OPEN ACCESS

University of BRISTOL

This electronic thesis or dissertation has been downloaded from Explore Bristol Research, http://research-information.bristol.ac.uk

Author:
Pickard, Nigel Brougham

Title:
The development of fuzzy knowledge base for machine monitoring.

General rights
Access to the thesis is subject to the Creative Commons Attribution - NonCommercial-No Derivatives 4.0 International Public License. A copy of this may be found at https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode This license sets out your rights and the restrictions that apply to your access to the thesis so it is important you read this before proceeding.

Take down policy
Some pages of this thesis may have been removed for copyright restrictions prior to having it been deposited in Explore Bristol Research. However, if you have discovered material within the thesis that you consider to be unlawful e.g. breaches of copyright (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please contact collections-metadata@bristol.ac.uk and include the following information in your message:

• Your contact details
• Bibliographic details for the item, including a URL
• An outline nature of the complaint

Your claim will be investigated and, where appropriate, the item in question will be removed from public view as soon as possible.

THE DEVELOPMENT OF A FUZZY KNOWLEDGE BASE FOR MACHINE MONITORING

By

Nigel Brougham Pickard.

A thesis submitted to the University of Bristol in accordance with the requirements for the degree of Ph.D. in the Faculty of Engineering, Department of Engineering Mathematics.

September 1989

Signed .....

Date 21/9/89

ABSTRACT

This thesis describes work done as part of a PhD CASE studentship in the Department of Engineering Mathematics at the University of Bristol, in cooperation with the South Western Regional Headquarters of the Central Electricity Generating Board.

This work follows on from that done at Bristol by a previous CASE student.

The initial aim was to "Develop a fuzzy knowledge base for machine monitoring". The machine referred to is the mains turbo-generator, as is operated by the CEGB in power stations throughout the country.

Discussion with experts at the CEGB and analysis of case histories indicates that domain experts often perform reasoning under high uncertainty, to deduce fault conditions from machine vibrations and choose remedial actions.

A new model of uncertainty has been devised which is robust under the adverse conditions of high uncertainty which characterise the problem domain. Realistic evidence for conclusions can be derived on the basis of uncertain expert knowledge, even when many symptoms are unknown.

This has been the basis for an expert system implemented in PROLOG. The form of the new representation allows the expert to perform knowledge acquisition simply and to give commentary on its reasoning.

It is recognised that in order to make the system more autonomous, the present knowledge representation should be incorporated within a temporal framework. Also, the expert system would benefit from being reimplemented in a language such as FRIL which would enable a more flexible user interface to be written.

ACKNOWLEDGEMENTS

I am grateful to my supervisor, Dr Bruce Pilsworth, for his help and encouragement during my studies at Bristol.

I would like to thank all those past and present of the ITRC at 93 Woodland Road, especially Dr Trevor Martin for his cheerful optimism and advice in numerous matters.

Last but not least I shall thank Helen for her unfailing support.

MEMORANDUM

The accompanying dissertation entitled "The Development of a Fuzzy Knowledge Base for Machine Monitoring" is based on work carried out by the author at the University of Bristol between October 1984 and September 1987.

All work and ideas in this dissertation are original unless otherwise acknowledged in the text or by reference.

Signed .....

Date 21/9/89 .....

TABLE OF CONTENTS

FIGURES

CHAPTER 1

EXPERT SYSTEMS AND KNOWLEDGE REPRESENTATION

1.1 About this thesis

1.1.1 Historical overview of thesis

This thesis is based on work done in the Engineering Mathematics Department of the University of Bristol as part of a PhD CASE studentship.

The work was carried out in cooperation with the South Western Regional Headquarters of the Central Electricity Generating Board (CEGB).

The initial aim of the studentship was to:

"Develop a fuzzy knowledge base for machine monitoring."

The machine referred to in this title is the mains turbo-generator.

The mains turbo-generator is a large and very complex machine used by the CEGB to generate electrical power for the mains from hot pressurised steam created by various types of furnace.

The turbo-generator work followed on from work also done at Bristol, by Laurie Pegrum as part of an earlier CASE studentship. The net result of Pegrum's work was an IBM PC computer language called FRIL (Fuzzy Relational Inference Language) [Baldwin, 1983] which was based on Zadeh's theory of fuzzy sets [Zadeh, 1965] and extentions by Baldwin [Baldwin,1982]. Pegrum also produced a small knowledge based system written in FRIL for the diagnosis of major faults on CEGB fans. These are machines used to supply cooling hydrogen to turbo-generators.

Unfortunately, when Pegrum finished his three year CASE studentship he lost commitment to his PhD studies and his work was never fully written up.

The starting point for the turbo-generator work was the PC based FRIL language and the simple 'fans fault diagnosis system'.

The initial stages of the project concentrated on gaining a good appreciation of the generic turbo-generator and of the scope of the monitoring problem especially from the point of view of diagnosing faults on the basis of information about machine vibrations.

The main sources of knowledge and expertise were a group of experts at the CEGB's SW Regional HQ and a number of fault reports which they supplied. These are documented historical incidents in which abnormal machine behaviour was reported and the diagnoses that were produced.

The experts in this case were effectively trouble shooters for surrounding power stations. If a problem was suspected of a turbo-generator in a power station then these experts were informed.

Available information would be considered, and if necessary tests performed. The experts would then decide what action if any needed to be taken and recommendations would be made.

As project work proceeded it emerged that the fuzzy set theory representation was inadequate for the task of implementing turbo-generator fault diagnosis. The representation was not sufficiently expressive to represent the wide ranging probabilistic uncertainty and vagueness in knowledge used by the experts and data on which diagnoses were often based.

Additional problems arose with the PC based FRIL language [Pegrum, 1984] to do with the inbuilt programming environment, bugs in the language, inadequacy of documentation, and program size limitations. These will be described in chapter 6 on expert system implementation and design.

During the second year, a more flexible knowledge representation was devised which could represent uncertainty in a more accurate and concise manner. The knowledge which had so far be acquired was reformed and a new expert system produced by implementing an interpreter using the PROLOG language

PROLOG [Clocksin & Mellish, 1981; Clark & McCabe, 1984] is a popular artificial intelligence language based on first order predicate logic and the resolution method of inference.

Knowledge acquisition continued throughout the next year or so. Knowledge was transferred to the expert system which was enhanced by adding a flexible user interface (or front end), and knowledge acquisition and explanation facilities.

During this time some of the CEGB experts used the developing expert system in order to validate it. The CEGB experts were quite surprised by the system's ability to correctly diagnose faults from the symptoms they supplied. A number of historical scenarios were posed to the system which performed well without exception.

The following section describes the organisation of the remainder of the thesis in terms of the 7 chapters and their contents.

1.1.2 Organisation of material

The thesis is divided into six chapters including the present one which is introduced in the following section 1.2.

Chapter 2 describes the problem of machine monitoring for turbo-generators from the practical point of view.

Chapter 3 discusses reasoning under uncertainty. It looks at several models of uncertainty including probability theory, Zadeh's possibility theory, Shafer's evidence theory, and a newer evidential model of uncertainty which was developed at Bristol University by Jim Baldwin.

Chapter 4 concentrates on a method of knowledge representation and inference which was developed as a solution to the problem of fault diagnosis for turbo-generators. The theory is described in detail and an extensive worked example is given.

Chapter 5 discusses knowledge acquisition which is the process by which knowledge is transferred to the expert system. The main emphasis is on manual knowledge acquisition which involves work by a knowledge engineer who interviews experts and sorts through case histories and documentation to determine the knowledge being used. This is then represented in a form which is consistent with the chosen system of knowledge representation and inference.

Chapter 6 explains the design and implementation of turbo-generator expert system and gives a brief user guide to it in order to illustrate the systems level of competence from a user point of view.

Chapter 7 describes possible further work.

There are two appendices to this thesis. Appendix A contains the final expert system knowledge base that was produced. Appendix B contains sample output from the expert system. This shows the system in action to emphasise and illustrate some of the points made in chapters 1 to 7.

1.2 Introduction to chapter 1

The remainder of this chapter is divided into 4 sections. The first of these, section 1.3 describes expert systems in general and their structure.

Section 1.4 describes knowledge from a general point of view, and section 1.5 describes various methods of knowledge representation.

Section 1.6 considers a small number of typical, expert system applications including fault diagnosis for turbo-generators.

1.3 Expert systems

1.3.1 What is an expert system ?

An expert system is a computer program that can act in a similar way to a human expert in a restricted domain of application from the point of view of solving problems, taking decisions, planning and giving advice.

The activity of developing expert systems involves the application of a variety of techniques from artificial intelligence, and it is commonly called knowledge engineering.

Expert systems differ from other kinds of AI program in the following respects:

1) expert systems deal with problems of realistic complexity that normally require considerable expertise,

2) expert systems emphasise domain-specific problem solving strategies in contrast to the more general 'weak methods' of AI,

3) expert systems must exhibit high performance in terms of speed and reliability in order to be useful tools,

4) expert systems must be capable of offering plausible explanations and justifications for their solutions and recommendations in order to establish the credibility of their reasoning and knowledge.

The kinds of knowledge encoded in expert systems include:

1) definitions of facts and theories as given in text books and reference books on the problem domain,

2) heuristic knowledge and rules of thumb that are commonly applied in the application of the specialist expertise. This type of knowledge is usually acquired through experience.

Conventional computer programs depend on complete analysis of all the elements and steps in a problem. In effect this limits the domain of conventional computing to problems that can be exhaustively analysed.

Human expert regularly tackle problems that are far too large and complex to be understood completely. Heuristics are incorporated in to expert systems to enable them to also solve large and complex problems.

Heuristics enable the human expert to make educated guesses when necessary, to recognise promising approaches to problems, and deal effectively with error prone or incomplete data. However, heuristics by their very nature, can lead to errors. They do not guarantee the correct answer; they only increase the likelihood of finding a usable answer.

Expert systems place greater emphasis on knowledge rather than formal reasoning methods. This is suggested by the following broad range of application areas:

1) Medical decision-making and management,
2) Stock market and financial planning,
3) Weighing the evidence in legal cases,

4) Fault diagnosis and machine monitoring,
5) Command and control problems,
6) Speech and vision understanding,
7) Modelling the dependability of software,
8) Improving man-machine interaction,
9) Machine learning systems.

The most interesting and difficult problems of this kind do not have tractable algorithmic solutions. They often originate in complex social of physical contexts which generally resist precise description and rigourous analysis. This gives rise to a need for representations of uncertainty, which are introduced later in this chapter and considered in detail in chapter 3 on reasoning under uncertainty.

1.3.2 The structure of an expert system.

An expert system comprises two main parts. One part is a knowledge base and the other part is an inference engine which allows queries to be answered by asking queries of the environment and performing evidential (or non-monotonic) reasoning.

An expert system will normally contain other parts to enable modification of its knowledge base and to facilitate communication between the expert system and human operators. It is especially important for the expert system to be capable of explaining its actions so that they can be monitored and understood.

The idea of separating knowledge and inference is central to the expert systems approach. It is reasoned that a well expressed problems and expert knowledge lead more naturally to solutions.

In conventional computer programs knowledge about a problem and procedures for manipulating that knowledge to solve the problem are mixed together.

It is often very unclear from conventional computer program code what the program knows or assumes about the world. This has a number of disadvantages. Firstly, it emphasises the importance of the programmer and ensures that the non-programmer, domain expert will not be able to play a full role in the system's development. Also, other experts may be unable to see what the developer has assumed about the program. This may have unfortunate implications for maintenance of the program.

A number of expert system 'shells' are available commercially. An expert system 'shell' is simply an expert system with an empty knowledge base. All other parts of the expert system, such as the inference engine, user interface, and explanation facilities are ready for use. To produce an expert system, the 'shell' is filled with knowledge according to a predefined knowledge representation such as 'horn clause logic'. The shell will normally be equipped with a knowledge acquisition module to allow easy transfer of knowledge in to the knowledge base.

Expert system shells are a useful and efficient route to generating an expert system, assuming that a suitable shell can be found. However, the advantage of having a partially ready made expert system is usually at the cost of reduced flexibility. Except for certain applications, it is unlikely that an expert system shell will be sufficiently flexible to cope with all aspects of a problem domain. In this respect it is preferable to write the expert system in a powerful AI implementation language such as PROLOG or FRIL (a probabilistic extension of PROLOG) which will described in chapter 3 on reasoning under uncertainty.

1.4 Knowledge

Knowledge is more complex than information, and more valuable. Normally the term knowledge refers to a body of information about a particular topic that is organised to be useful. For example, if an individual is very knowledgeable about a subject, it is assumed that he not only knows lots of facts about the subject, but also can use that information to analyse problems and make judgements about related topics.

AI researchers have focused on the verbal and graphic aspects of knowledge rather than the more mathematical aspects that have been the concern of conventional software programmers. Thus, where a conventional programmer might seek to reduce a problem to elements that can be represented in mathematical terms and manipulated by an algorithm, AI programmers are more interested in knowledge expressed in sentences and pictures and manipulated by logical inferences.

1.4.1 Knowledge Representation and Inference.

An important consideration in constructing an expert system, is the choice of knowledge representation. This must be chosen in the light of the type of information and tasks that are being performed.

As a rule of thumb, the most natural representation of knowledge will be most appropriate. For example, a logical 'rule based' representation is ideally suited to representing decision trees, but will most likely be unsuitable for representing pictorial information.

A natural representation will simplify knowledge acquisition, inference, and explanation. However, a knowledge representation must not be chosen in ignorance of the way in which the expert system is to behave.

For example, diagnostic knowledge that relates symptoms to illnesses may be represented using 'IF-THEN' rules of the form:

IF X THEN Y

where X and Y are, possibly compound, propositions that relate to a patient.

X could be either symptoms or illnesses, and similarly for Y. If X is symptomatic, then the rule will be data driven and will therefore be suitable for answering questions for the form:

'what illnesses Y are indicated by the symptoms X'.

IF Y is symptomatic then the rule will be problem or query driven and will therefore be suitable for answering questions of the form:

'are the illnesses X indicated by the symptoms Y'.

If expert knowledge derives from an expert then a natural or 'intuitive' knowledge representation will be easier to work with. Knowledge will be more easily represented in its natural form and the knowledge base will be easier to service.

An added bonus of having a natural form of knowledge representation is that the expert system can more easily produce explanations of its actions that are intelligible to a human operator.

1.4.2 Knowledge Engineering.

Knowledge engineering is performed by a knowledge engineer, and is the process of transferring expertise to the knowledge base of an expert system.

The knowledge engineer either elicits knowledge directly from a human expert or uses some means of inducing knowledge from examples or data.

The knowledge engineer may use knowledge elicitation tools to help to query the expert in an objective way and represent the expert's views in an intuitively meaningful fashion. Two such knowledge elicitation tools are 'repertory grids' and 'multi-dimensional scaling', both of which are described in Chapter 5 which deals with knowledge acquisition.

Having obtained some expert knowledge, the knowledge engineer then represents it in a form which is compatible with the expert system and its intended mode of use.

1.4.3 Uncertainty.

Chapter 3 explains the extensive problems of reasoning under uncertainty in reasoning about the behaviour and underlying condition of turbo-generators.

In this and many other applications, such as those listed in section 1.3.1, a method is needed for handling uncertainty.

prUncertainty can arise for the following reasons:

1) lack of expert knowledge about the domain of expertise,

2) lack of data, or knowledge of the problem,

3) poor definition of requirements.

Lack of expert knowledge is the most obvious cause of uncertainty. For example, it is essential for a motor mechanic to have a clear understanding of motor car engines in order to diagnose faults in them. As engine vary between makes of car, it may also be necessary to know the idiosyncrasies of particular engines.

Lack of data, or knowledge of the problem is another important source of uncertainty. If the problem is not adequately defined in terms that the expert can understand, then the chances of a solution are greatly reduced. For example, when taking a car for repair at a service station it is important to describe the problem as precisely as possible. In this way the mechanic can clearly recognise the symptoms and have a good chance of diagnosing the underlying fault. If the problem is poorly specified the mechanic may well home in on a general problem like 'poor engine tune' which may be justified but not be the source of the problem.

Poor definition of requirements, or an ambiguous query, is a source of uncertainty which is often overlooked. For example, if a customer takes his car in for an MOT, it is inadequate to leave instructions to 'make any necessary repairs'. The problem is partly that it is unclear what is 'necessary'. The aim may be to get the car through its MOT and sell it with minimum cost. On the other hand the customer may wish the car to last for years to come and will therefore tolerate greater expenditure on lasting repairs.

The three uncertainties described above are the root cause of uncertainty in reaching conclusions. Given one or more of these uncertainties, conclusions are bound to be uncertain to some degree. However, if reasoning is not performed carefully this 'inherent' uncertainty may be unnecessarily increased.

In chapter 3 on reasoning under uncertainty, several representations of uncertainty are considered.

In the next section a number of methods of knowledge representation are considered, but without special reference to uncertainty.

1.5 Methods of knowledge representation.

The following nine sections describe some of the better known methods of knowledge representation that have been either used in expert systems or have been identified as being potentially useful.

1.5.1 Logic.

Logic is the best known method of knowledge representation. It was originally devised as a precise and unambiguous method of representing factual information, to which formalised methods of reasoning could be applied. This is a principal activity of artificial intelligence and in particular, expert systems.

Logic in the form of first order predicate calculus is of particular relevance to expert systems. The rules of inference have been shown to be complete in the sense that they can derive all valid consequences of any set of premises.

The inherent modularity of logic is also of importance to expert systems as it eases knowledge engineering: knowledge acquisition and maintenance.

Logic is also a flexible representation that enables a variety of methods of reasoning. The disadvantage of this is that the form of representation does not clearly indicate the method of deduction that should be applied. This is a major problem of artificial intelligence, that the method of deduction becomes more open ended and therefore more complex as the representation becomes more flexible.

There have been attempts to incorporate heuristics in to the overall deduction process as a means of controlling deduction without compromising logical clarity and precision [Hayes, 1977; Weyhrauch, 1978].

A further attraction of logic is that it is an intuitively acceptable means of representing human knowledge [McCarthy, 1977; Filman, 1979].

Criticisms of logic have come from Minsky [Minsky, 1982] and others, who think that the best way to tackle AI problems is to mimic human reasoning, which they believe to be impractical using logic. This is essentially a criticism of the methods of deduction that are applied to logically represented problems rather than the representation itself.

The useful applications of logic as a means of representation are extended by applying forms of reasoning that suit thee problem domain. For example, non-monotonic, probabilistic and fuzzy reasoning schemes can be used to cope with uncertainty that can arise in a problem domain. These and other enhancements to logical reasoning are described later on in this chapter.

There is a fundamental objection to using logic to represent natural language as most commonly used connectives and qualifiers cannot be accommodated: 'Because', 'although', 'a few', 'sometimes', for example. Also there is no mechanism in logic to reflect the inherent differences between nouns, adjectives and verbs, so that 'the sky is blue', 'the sky lit up', and 'the sun rose in the sky' have to be represented in the form P(sky).

Countable and mass nouns such as 'watches' and 'butter' also present a problem as they are not individuals and so cannot be represented by constants.

A further major problem of logic is it can only represent propositions and is unable to represent imperative or interrogative language although some attempts have been made to devise 'imperative' and 'interrogative' logics.

However, from the point of view of constructing expert systems, many of the above objections are somewhat philosophical and do not detract much from the overall usefulness of the representation.

1.5.2 Modal Logics.

The family of modal logics extend predicate logic by introducing symbols M and L which can prefix any valid formula to give other valid formulae.

One interpretation of M and L is such that if p is a valid formula then Mp and Lp represent 'p is possible' and 'p is necessary'.

In the light of this interpretation of M and L, suitable axioms are added to those of first order calculus. For example, 'Lp --> p' which means that 'if p is necessary then p is true'.

Another interpretation of M and L is such that Mp and Lp represent 'p is believed' and 'p is known'. This has useful applications in AI planning systems [More, 1984; Konolige, 1982].

The idea is that things which are not yet known, but which are expected to become known (possibly as a result of later actions), are assumed or 'believed'.

This avoids the problem of logical inconsistency which arises in classical logic when an default assumption is shown to be false. It is logically possible for something which is believed to be shown to be false, but it is not possible for an assumed fact to be shown to be false.

However, these approaches are limited because it is only acceptable to 'believe' something which it is planned to establish the truth of. This is a serious limitation in diagnostic problems, for example, as the receipt of information can not always be guarantied.

A solution to this problem is provided by a non-monotonic approach such as Doyle's 'truth maintenance' [Doyle, 1979].

Support logic can deal with this problem by associating appropriate evidential uncertainty with any assumptions that are made. Later, possibly conflicting, evidence may override the assumption without any inconsistency arising. An illustration of this capability is given in the section on non-monotonic logic which follows shortly.

1.5.3 Multi-valued Logic.

Multi-valued logics are another attempt to deal with the problems of reasoning on the basis of incomplete or uncertain knowledge [Rescher, 1969; Haack, 1974].

A system of multi-valued logic was designed by Kleene [Kleene 1952] to capture a state of partial ignorance. A third truth value 'unknown' was added to the usual 'true' and 'false', and was applied to any statement which could not be shown to be definitely true or false.

However, this method is unsuitable for the task of making assumptions and later rejecting them if necessary. The approach is too blunt and simple.

Other systems of multi-valued logic have been devised [Lukasiewicz, 1920; Bochvar, 1939] but are even less suitable than that of Kleene.

1.5.4 Non-monotonic Logic.

A non-monotonic system of reasoning is characterised as follows: given any set A of axioms which lead to a certain set C of conclusions, enlarging A to A' by adding some further axioms may lead to both, further conclusions not deducible from A being reached, and some of the conclusions in C being discarded. This contrasts with standard predicate logic where it is always true that, if A is a set of axioms, and F is any formula such that A |- F, ie. F is provable from A, and A is a subset of A' then A' |- F.

Non-monotonic systems of inference are of interest because they attempt to capture in a precise, formal way an aspect of the sort of practical reasoning that is required in real-life situations in general and expert systems in particular. Such reasoning must often be undertaken without full knowledge of all the facts relevant to the situation under consideration. In order to come to any conclusions at all, these gaps must be filled in by tacit or explicit assumptions that are reasonable in the light of our present incomplete knowledge. Later additions to our information may reveal that some of these assumptions were in fact false, and conclusions that were reached on the basis of them will then have to be discarded in the light of our fuller knowledge; hence the non-monotonicity.

Some systems of non-monotonic inference, rather than using default assumptions, have an operator M that can be prefixed to any proposition p, with Mp being interpreted as meaning 'p is possible' or, more formally 'p is not provable from the axioms'. The actual role of such formulas is related to that of explicit default assumptions, since they are mainly used to say that, in effect, 'As long as we do not know that p we will provisionally assume that p'. Such use can be expressed by axioms such as:

Mp --> p,

or

p^1 & p^2 & p^3 &....& p^n & Mp --> p.

The following example is frequently used as an illustration. In general, birds fly. which suggests the axiom:

All x. bird(x) --> flies(x).

However, some birds such as penguins, cannot fly, so this axiom could lead to false conclusions. In a non-monotonic system one could adopt the following version of the axiom:

All x. bird(x) & M(flies(x)) --> flies(x).

The false conclusion that Rhoda the penguin can fly would be blocked by the further axioms:

All x. penguin(x) --> not flies(x)
penguin(Rhoda).

These give 'not flies(Rhoda)' which will block the premise 'M(flies(Rhoda))' in the first axiom.

Various approaches have been devised to embody the principal of non-monotonic reasoning. One of the best known and ambitious approaches to incorporating non-monotonic reasoning in a formal system is that of McDermott and Doyle [McDermott, 1980; Doyle, 1982]. However, these systems are quite complicated and difficult to use. In them, the language consists of the usual symbols of predicate logic plus a symbol M as described above, that can be placed in front of propositions. Intuitively, for any proposition p, Mp is meant to mean 'p is consistent', ie. that 'NOT p cannot be proved from the current set of axioms. Thus the meaning of M in any particular context is defined relative to the set of axioms being used. However, if this set of axioms contains formulas which also involve M, then there is a circularity in the definition, and the meaning of M becomes unclear. The problem of giving a precise interpretation to M is not resolved by Mcdermott and Doyle's section on models, where a definition of what it is for a formula of the form Mp to hold in a model is simply omitted. M. Davis [Davis, 1980] has observed that there seems to be no way of repairing this omission.

The form of non-monotonic reasoning which is most likely to be of use in expert systems is that of 'truth maintenance systems' [Doyle, 1979]. The basic idea of such systems is to keep a record of the reasons for reaching any conclusion; then if an inconsistency appears, dependency directed backtracking is performed to locate the premises which underlie the inconsistency. The responsible premises are then retracted and the reasoning process is restarted.

Dependency-directed backtracking could be implemented in PROLOG by keeping a list of the successful goals involved in diagnosing a fault, for example. This type of information is also of use in explanation analyses, because it is comprises all the reasons for reaching the conclusion.

1.5.5 Temporal Logics.

Temporal reasoning is essential to many areas of human expertise, and is therefore an important consideration for expert systems.

In carrying out a diagnosis, the length of duration of events and symptoms, and the temporal relationship between them is a primary consideration.

Time is also an important factor in implementing cures or remedial actions. For example, 'if the carburettor floods then release the accelerator pedal, push the choke lever in, wait for 5 minutes, and try again'.

This could be represented in conventional logical terms by:

$$p_0(t_0) \text{ --> } (t_0<t_1) \;\&\; p_1(t_1) \;\&\; (t_1<t_2) \;\&\; p_2(t_2) \;\&\; (t_2<t_3) \;\&\; p_3(t_3) \;\&\; (t_3<t_4) \;\&\; p_4(t_4)$$

where $p_i(t_i)$, i=1..n, represent the propositions:

i=0 : 'the carburettor floods at time t_0'

i=1 : 'release the accelerator pedal at time t_1'

i=2 : 'push the choke lever in at time t_2'

i=3 : 'wait for 5 minutes (starting) at time t_3'

i=4 : 'try (to start the car) again at time t_4'

The propositions $(t_i<t_{i+1})$, i=0..3, ensure the correct temporal relationship between events, ie. that the remedial actions are carried out in the correct sequence.

This technique for representing dynamic behaviour is very powerful and does not reduce logical clarity or precision, but suffers from the proliferation of extra time variables and quantifiers.

Various systems of temporal logic have been devised [McArthur, 1979; Rescher & Urquhart, 1971] which divide time into past, present and future, using temporal operators which can be prefixed to sentences to yield new sentences. For example, for a sentence s, Fs and Hs mean respectively 's is true at some future time' and 's has been true at all past times'. However, the application of temporal logic to AI problems has tended to use an approach which uses primitives such as 'event', 'time interval', 'instant', 'state of the world', etc. [McDermott, 1982; Allen, 1981; Turner, 1984; van Benthem, 1983]

The planning method of Allen and Kooman [Allen & Kooman, 1983] could be applied to planning sequences of actions to deal with complex fault situations. This method can take into account events that overlap in time, unlike some methods in which events must occur in linearly ordered, non-overlapping intervals.

Moszkowski [Moszkowski, 1986] describes a temporal logic interpreter called 'Tempura, which implements 'interval temporal logic' [Moszkowski, 1985]. This system of logic incorporates conventional temporal operators such as 'next' and 'always', and lesser known operators such as 'chop'.

Essentially, the period of time which is of interest in a problem, is considered as a sequence of regular states in which each logical statement is either true of false. Each state is viewed as a snapshot of the world. Then for example, two events overlap in time if they are simultaneously true in at least one state.

'Tempura' is best suited to reasoning in areas such as computer programs, digital circuits, etc, where states can be easily identified, for example, by a certain number of CPU clock cycles.

In medical or machine fault diagnosis there are no regular measurable states and therefore temporal operators such as 'next' are meaningless because there is no way of knowing when the next state begins and ends.

A possible solution to this problem might be to define a unit of time of suitable length for each problem domain. The difficulty here is that in medical diagnosis,

for example, events will occur on a variety of time scales. Some bodily responses to a treatment will be very fast and others very slow. It would appear necessary that a chosen time scale must be small enough to clearly represent the faster changes, but this would lead to over precision on the slower changes.

It seems likely that an interval based approach to diagnostic problems will be most successful, but a constant time-stepping inference method would appear to be inadequate.

1.5.6 Production systems.

Production systems is one of the earliest AI techniques for representing and reasoning with knowledge. Production systems were based on the work done by Emil Post (Post 1930) and others, in attempting to model human problem solving.

Production systems comprise three distinct components:

1) a set of production rules,

2) a working memory or database, and

3) a rule interpreter.

Each production rule comprises a condition/action pair.

For each given condition, the rule interpreter searches through the production rules until it finds a match. It then carries out the corresponding action which could, for example, result in another condition for which the rule interpreter would have to find a match, and so on. Typical actions include, adding or deleting an entry in the database, sending and receiving messages, etc.

Part of the job of the rule interpreter is to choose the order in which the production rules are searched. Typically this defaults to scanning the rules top-down and left-to-right, and sometimes heuristics are incorporated into the search strategy to improve its speed and efficiency.

Commonly used heuristics include:

1) specificity

The more restrictive rules are used before the more general ones.

2) priority

Rules are ordered according to priority, so that high priority rules are encountered first.

3) size

The rule with the greatest number of conditions to match is used first.

4) concurrency

The most recently used, matching rule is reapplied. This is applicable to tasks which incorporate repetitive elements.

5) context

Rules are ordered in the rule base according to their context.

Production systems can be either goal (or hypothesis) driven by backward chaining to data, or data driven by forward chaining to solutions.

Production systems have been popular in AI mainly because the simplicity of the rule interpreter and the modularity of their rule bases. This means that rules can be added to a rule base without need to change the interpreter or existing knowledge.

The draw back of the method is that somewhat adhoc heuristic methods may be used in the search strategies for rule interpretation and incorporation of uncertainty in practical implementations.

1.5.7 Semantic networks.

Semantic networks were first proposed as a method of knowledge representation in the late 1960's as a means of natural language understanding by Schank, Woods and Wilks [Schank, 1975; Woods, 1975].

A semantic network is a directed graph, consisting of nodes which represent concepts, and labelled directed arcs which represent relationships between concepts.

By skillful choice of concepts and relations, very complex groups of interrelated facts can be represented.

The 'is-a' relation of special importance as it indicates membership of a node to a particular class, and is a powerful method of building up inheritance hierarchies. This enables efficient storage of information, and in particular, defaults.

Information that is not directly associated with a node may be accessed through inheritance. However, this can lead to inconsistencies where there is more than one 'is-a' link. For example, a pen is a writing implement but it is also a type of bird (a female swan in fact). To overcome this problem, preferences may be needed where there is more than 'is-a' relation associated with a concept.

Another problem with accessing default information through inheritance is that after a number of stages of inheritance the information obtained can become tenuous. To reduce the likelihood of problems occurring, a breadth-first strategy is used for searching through the hierarchy. This ensures that the most closely related concepts are encountered first through the type hierarchy.

A problem of knowledge representation using semantic nets is representing the quantification of each node and its scope - this is much more natural in predicate calculus.

Other problems include the vagueness of the distinction between concepts and relations, and the fact that in practical contexts, the graphs can become very large and complex. Also, the meaning of graphs can become unclear without considering the procedures which are used to access and interpret the graphs. This encourages an undesirable ad-hoc and informal approach to knowledge representation.

1.5.8 Frames.

Frames were proposed by Minsky [Minsky, 1974] as a form of knowledge representation. A frame consists of a body of declarative procedural information.

A frame is initially conceived as a set of empty slots, each waiting to be filled with information that is appropriate to its type. Information derived using external knowledge or by default information stored within the frame. Slots may be frames themselves, so that complex knowledge can be neatly represented.

For example, a persons vital statistics could be represented using a frame consisting of slots for name, age, height, weight, and various other details. There might also be slots containing procedures for calculating height from weight if the height were unknown but the weight were known. There might also be a slot containing a default (eg. an average) height to be used if there were no information about height or weight.

Frames are similar to semantic networks but suggest a higher order and more modular representation where whole clusters of nodes and links are stored.

In general, the appearance of information about a concept will depend on which frame through which information is accessed. This enables the frames to highlight different perspectives of information about concepts.

1.5.9 Conceptual Graphs.

Conceptual graphs are an amalgam of ideas from frames and semantic networks but with more formal and rigourous semantics which makes it amenable to implementation as a computer language.

The name and interpretation of conceptual graphs comes from Sowa [Sowa, 1984].

Conceptual graphs consist of two types of nodes: concepts and relations. A single concept on its own is a valid conceptual graph, but each conceptual relation must be attached to some concepts.

Conceptual graphs differ from semantic nets in that each conceptual graph represents a singe proposition, and each concept node comprises two parts: a type label, and a referent field.

The type label of a concept node is analogous to the 'is-a' link in semantic nets but gives rise to a clearer and more modular representation.

A semantic network could be interpreted in conceptual graph terms as a set of definitions of types and relations in conjunction with a type lattice that gives an interpretation on how different types are related.

A concept label can be defined using either 'genus' or 'schema'.

Genus defines a concept's type by characterising the differences between the concept and a more general concept, whereas schema uses a collection of various examples of the concept.

The language of conceptual graphs is useful in natural language modelling because it has a high level representation of knowledge and rich semantics for modelling context and pre-suppositions.

1.6 Applications of expert systems.

This section describes a small number of expert systems applications, one for computer system configuration and others for monitoring and diagnosis.

1.6.1 XCON.

XCON is an expert system for configuring the Digital Equipment Corporation's (DEC) VAX-11/780 computer systems. [Harmon & Maus & Morrissey, 1988]

A customer's order is input to XCON by means of a dialogue. XCON then determines the exact specifications and layout of the components required to meet the order, and outputs a set of diagrams displaying the spatial relationships between components. These diagrams are used by technicians who physically assemble the system.

XCON facilitates DEC's policy of offering a wide selection (420 in 1979) of computer hardware components for the customer to choose from. DEC prefers this policy to marketing whole preconfigured systems. In particular, XCON has allowed DEC to reduce the size of its technical editor staff and has allowed the

remaining editors to focus their attention on exceptional tasks that the system cannot handle.

XCON uses a rule based knowledge representation supported by the OPS5 environment which is a development from the earlier OPS4 in which it was originally developed.

The initial prototype of XCON was engineered by John McDermott and Carnegie-Mellon University. This was done over a period of 3 man months and resulted in a system containing 250 backward chaining rules. This was developed in to the first full, 750 rule version of XCON, in just one man year.

This development was effected by using case studies. The system was given a series of problems, the solutions to which were evaluated by experts. If the answers given by the system were judged to be inappropriate then additional rules were added so that the system would answer those problems correctly in the future.

In June 1981, further to validation and various enhancements which reduced the size of the rule base, the system was put into place in all manufacturing facilities of DEC. It has been in use and maintained since then.

In spite of the success of the XCON system, is interesting to note the view of John McDermott, that had the target for the expert system been the PDP-11 rather than the VAX-11/780 then the project would probably have failed. This was due to the limitations of the original OPS4 system used for implementation.

1.6.2 IDEA.

The IS4000 Diagnostic Expert Adviser (IDEA) is an 'in-house' development of Pacific Bell's Expert systems Projects Group, and took six months and 1.5 man years of effort to produce. [HARMON & MAUS & MORRISSEY, 1988]

IDEA is a stand-alone PC based system which assists telephone technicians in diagnosing problems with Infotron IS4000 Local Area Network, a complex switching device.

Input to the system, which takes under 5 minutes, is carried out by a user who selects from menus that best describe the problem situation being encountered.

Based on early answers, the system asks further questions until is has enough information to recommend an action.

IDEA was developed by an internal knowledge engineer, using an experimental Pacific Bell PC product called MetaShell, a high level environment which controls interactions between 19 Exsys subdomain modules (Exsys is essentially a restricted version of EMYCIN). These modules are reusable in the design of other diagnostic systems servicing similar hardware domains. [HARMON & MAUS & MORRISSEY, 1988]

Features of the IDEA expert system include on-line help, information sharing between knowledge bases, unlimited rule capacity due to modular design, and a comprehensive natural language audit trail (or explanation).

1.6.3 DELTA.

The Diesel-Electric Locomotive Trouble-shooting Aid (DELTA), formerly known as the Computer Aided Trouble-shooting System (CATS-1), was developed by the General Electric Company (GEC) in Schenectady New York, to help railroad maintenance personnel to maintain their diesel-electric locomotives. It allows them to diagnose maintenance needs and prescribe maintenance action. [HARMON & MAUS & MORRISSEY, 1988]

DELTA uses a rule based form of knowledge representation and a hybrid forward/backward chaining inference strategy. This strategy, together with trouble shooting rules, is used to isolate faults and to generate enquiries. In addition there is a help system that uses a forward chainer together with a rule based engine taxonomy responds to user requests for information such as the location and identification of individual locomotive components, replacement parts classification, and description of repair procedures.

Users operate the system via a computer terminal, at which they are presented with a menu of possible fault areas. When an area is selected, the system proceeds with a detailed series of questions. For example:

"Is the fuel filter clogged?"
"Are you able to raise the fuel pressure to 40 PSI?"

The answers to these questions are used to generate recommendations via rules such as:

"If engine_set_idle and fuel_pressure_below-normal and fuel_pressure_O.K., Then fuel_system is faulty.

DELTA also interfaces with a device that allows the system to print out diagrams, and a video disk player that allows the system to display diagrams to show where particular components are located on the locomotive. If requested, DELTA can also display training film sequences to show the user exactly how to make a particular repair.

DELTA was originally developed in LISP but was later converted to FORTH, a portable language easily adaptable to any microprocessor.

The source of (rule based) expertise for DELTA was a number of maintenance experts, although most work was done with a senior field service engineer of 40 years experience and an acknowledged expert on the field of maintenance for diesel-electric locomotives. The overall development took from 1981 to 1984 and involved a team of knowledge engineers. This resulted in a knowledge base for the production prototype, of some 1200 rules.

1.6.4 EXTASE.

EXTASE [Jakob, Suslenschi, Vernet, 1986] is an expert system for alarm processing in the vacuum distillation tower of a refinery. The system is able to find the causes of active alarms and the dependencies between alarms that originate from the same process disturbance. The system offers a framework for hypothesis driven reasoning, and has a degree of tolerance to sensor failure.

Modern chemical plant processes are increasingly being operated through computerised equipment that collects information measured on the system. When control parameters exceed preset thresholds, operator attention is requested by built in "alarm signals".

The operator must use his skill and experience to deduce the underlying cause(s) of the various alarm signals.

The process is complicated by several factors:

1) the available information is not be completely reliable because of possible sensor failures,

2) the order in which alarms are signaled depends in part on the values of process thresholds and therefore the alarms which are signaled first are not necessarily closest to the origin of the problem,

3) the need for a prompt response to alarms can restrict the operator to a shallow and often incorrect analyses.

The reasoning of the operator is thought to be hypothesis driven. When alarms are signaled, the operator hypothesises underlying causes. The operator then uses his knowledge of the causal relationships between the process operation and process control values to infer the alarms that would correspond to the hypothesised causes. If these are consistent with the actual alarms then hypotheses are considered to be worth pursuing.

The EXTASE expert system represents operators' causal knowledge using causal relationships referred to as 'couplings'. For example there is a coupling between the incoming flow to a tank, and the level of fluid within the tank.

Couplings are represented by means of production rules. However, this is not fundamental to the design of the system, but is a convenient means of implementation.

Rules are of the form:

IF (X1 AND X2 AND ... AND Xn)
THEN (Y1 AND Y2 AND ... AND Ym)

where the Xi (i=1..n) and Yi (i=1..m) are process conditions.

In order to generate hypotheses about possible faults, the system backward chains through the rules, and then tests the hypotheses by forward chaining through the rules and comparison with alarm readings.

The system does not consider historical data and only uses snapshots of actual process values. In addition the system is not connected directly to the process & therefore requires operator input which must approximate process values to a five point scale from 'very low' to 'very high'.

The latter would seem to be significant in a problems of this kind which deal with large numbers of continuous, interacting variables. The ranking scales have an implicit context which must be taken into account. In the context of a large shallow tank, 'very high' could mean '1 metre higher than normal', whereas in the context of a large tall tank, 'very high' could mean '10 metres higher than normal'. This could also cause problems for operator who has to give subjective rankings for process values, and for the knowledge engineer who has to construct a consistent causal model of the process.

It seems likely that a slightly more fundamental model of the process and plant would be better in the long run. Information about the state of the process could be represented using approximate volumes, flow-rates, temperatures, etc. Given these, and knowledge about the dimensions and other properties of the plant, fluid levels and other parameters could then be calculated. The advantage of this approach is that the interaction between complex systems becomes more implicit and obviates the need for contextual complications.

The representation of time is also somewhat restricted. EXTASE allows for rates of change to be represented, but for example, cannot take into account factors such as propagation delays.

On the positive side, EXTASE supports a backtracking search mechanism which enables a form of truth maintenance (Doyle,79). For example, it may be initially assumed that a process reading is correct. If it later proves to be false then backtracking occurs to just before the point at which the false reading was used.

1.6.5 An expert system for fault diagnosis on turbo-generators.

This section briefly describes an expert system for diagnosing faults on turbo-generators, written for the CEGB by GEC Research and the UK branch of Carnegia, producer of the Knowledge Craft development tool.

The expert system as described in 'Expert Systems User' magasine [Roth & Bannister, 1988] appears remarkably similar in its approach to that already developed by the Author and which is described in this thesis.

The GEC produced expert system is based a logical knowledge representation consisting of supported facts and rules and uses a generate and test strategy for fault diagnosis.

The exact mode of input to the expert system is unclear, but it appears from the description in Expert System User that input is currently manual but based on data obtained from a 'level 2' system which in turn receives input from a 'level 1' system.

The 'level 1' system comprises Vibration Monitoring Equipment (VME) which, when installed on a machine, takes vibration readings and records them using analogue multichannel tape recorders. The recorded data are analysed off-line, and can be extracted and presented graphically. If vibration levels exceed certain thresholds, alarms are automatically triggered.

The 'level 2' system performs more detailed analysis of the data produced by the 'level 1' system. This is needed because a turbo-generator's vibration pattern can be affected by its operating conditions. The 'level 2' system has to extract these effects so that the underlying trend in vibration patterns can be seen. This is done by complex trend and statistical analysis.

The 'level 3' expert system takes over where the 'level 2' system leaves off and is aimed at improving the experts' performance in diagnosing faults.

The expert system can generate hypotheses to explain abnormal machine behaviour and then rate the hypotheses according to the weight of available evidence. There is also a facility within the system to set a confidence threshold so that hypotheses with low confidence will not be pursued.

The GEC 'level 3' expert system work was only in its 'feasibility study' phase during the time at which the author had most contact with the CEGB. Furthermore, the author was not given official access to the results of the feasibility study, but was allowed a very brief unofficial browse of the feasibility study report.

Uncertainty seems to have been represented using a single certainty factor, similar to that used by MYCIN, which is attached to facts and rules. The certainty factor can take values between 1.0 and -1.0, such that certainty factor of 1.0 indicates conclusive supporting evidence, and a certainty factor of -1.0 indicated conclusive opposing evidence.

This is clearly less flexible than the Support Logic approach (see chapter 3) of using a support pair consisting of two numbers which represent the lower and upper bounds on the probability.

In the -1.0 to 1.0 uncertainty scheme, situations of equal supporting and opposing evidence and no evidence at all are represented by identical certainty measures, ie. 0.0. From the point of view of fault diagnosis this may at first appear reasonable, but gives misleading results when used to calculate expected utilities as a basis for choosing remedial actions.

The GEC expert system's knowledge base contains both surface and deep knowledge.

The surface knowledge was induced from sets of examples and counter examples using inductive software produced by Carnegia, and then modified before incorporating it into the knowledge base.

The surface knowledge on its own was not considered sufficient for fault diagnosis as it would be too 'brittle' to cope with unusual data, and would produce obscure explanations of its reasoning.

The deep knowledge consists of a causal network or lattice similar to that used for hypothesis refutation in the author's expert system. This gives a more complex model of the task domain, providing an underlying model of the more fundamental concepts involved.

The CEGB view their new expert system as a means of improving their experts' performance in fault diagnosis, by making available a large body of expertise and data which can be easily accessed. In this way they believe that diagnoses can be made more quickly, and in view of the huge loss of revenue associated with machine down time, they believe that the system's will rapidly pay off its development and installation costs.

This concludes chapter 1. The following chapter describes the problem of turbo-generator fault diagnosis in detail. Subsequent chapters are on reasoning under uncertainty, knowledge representation, knowledge acquisition, design and implementation of the expert system, and conclusions and further work.

CHAPTER 2

THE REAL PROBLEM OF STATE MONITORING FOR TURBO-GENERATORS

2.1 Discussion of state monitoring.

Turbo-generators are the machines used in oil, coal, and nuclear power stations to produce electrical power from hot, pressurised steam.

The turbo-generator comprises a number of connected units including four steam turbines, an electrical generator, and an exciter. Each unit has a central shaft which is coupled to those of its neighbours and thereby forms a straight, piecewise continuous central shaft.

The units are arranged in the following order along the machine:

1) high pressure turbine,
2) intermediate pressure turbine,
3) low pressure turbine 1,
4) low pressure turbine 2,
5) electrical generator,
6) exciter.

Hot pressurised steam is fed through the turbines, and causes the machine's central shaft to rotate. This causes the generator's windings to rotate within an electro-magnet which is energised using direct current produced by the exciter.

The turbo-generator is typically about 50 meters long and weighs several hundred tons. The power output from a single machine can be as much as 660 MW (660×10^6 Watts) which could, for example, satisfy the requirements of a city the size of Bristol.

The capital costs of replacing or repairing a typical turbo-generator are enormous, and the operating costs are very high.

The high costs associated with the turbo-generator necessitate careful monitoring of its health and performance. The monitoring strategy must consider and balance these three things:

1) the loss of profits that are being suffered due to poor performance, and the loss of profits that will be suffered if the machine has to be taken out of service for repairs,

2) the possibility that continuing to run a machine will result in damage to the machine that will be costly to repair,

3) the possibility that the machine may fail catastrophically with disastrous consequences.

The machine monitoring strategy seeks to minimise the overall cost associated with operating the turbo-generator. As the costs at stake are so high, it is worth taking great care in performing the machine monitoring efficiently so that the costs can be best minimised.

For many years machine monitoring has been performed by human experts, only recently have expert systems been a feasible alternative.

Experts in turbo-generator monitoring are people with considerable experience of turbo-generators, the faults that they can develop, and possible remedies.

Turbo-generator behaviour is very complex due to its large number of components, and the way that these components are linked together.

The turbo-generator and its foundations have a significant flexibility, and the whole arrangement vibrates in a highly coupled manner.

To determine the state of the machine from its behaviour is therefore very difficult.

It is not technically feasible to produce an overall deterministic model which links vibrations to the underlying structure of the machine. For example, two machines of the same design will have vibrational behaviour patterns that are essentially similar but different in important respects.

The detailed vibration patterns are due to aspects of the machine's structure which are beyond practical measurement due to the number of measurements that would be required and the difficulty in obtaining them. Also, the minute structure of the machine is continuously changing due to wear, metal fatigue etc.

Even if the minute structure of the machine could be determined, it would unfeasible to compute the relationship between the underlying structure of the machine and the machine's vibrational behaviour. Apart from these difficulties, the real need in state monitoring is to determine the structure of the machine on the basis of its vibrations. If the structure of the machine were known, vibrations would be of relatively little interest.

In view of the above difficulties in producing a deterministic model of the turbo-generator, a qualitative model must be used.

In spite of the complexity of the machine, certain facts about the machine tend to hold with varying degrees of certainty.

Human experts frequently reason successfully on the basis of such, uncertain knowledge. Indeed, for many years turbo-generator state monitoring has been performed by human experts using qualitative reasoning.

In order to devise an efficient computerised machine monitoring system it is necessary to construct an abstract framework for describing the turbo-generator and its behaviour.

It is helpful to conceptualise the turbo-generator as a system, and use some of the ideas from systems theory.

Definition of system:

A system is a complex whole, comprising a set of connected elements. The behaviour of the whole system, derives from the behaviour of its elements which interact with each other. The whole system can exhibit behaviour which is far more complex than the behaviour of any of its parts.

For example, a car can be described a system for transporting people. The car comprises many thousands of parts which are connected to form a whole. Each part connects to a number of other parts in certain manner so as to enable the car to transport people in an efficient manner.

Each component of the car behaves simply, but the whole car has complex behavioural characteristics. For example, the spring and shock-absorber assemblies are simple devices that reduce the amplitude and frequency of mechanical shocks between the car's wheels and 'chassis'. The behaviour of each assembly is simple to predict. However, much more complex behaviour can be seen from the car, when it is driven over a rough road.

There are three concepts that are central to systems theory that must be defined before proceeding with the analysis. The three concepts are:

1) system state,
2) system observability, and
3) system controllability.

System state:

The system state specifies the condition and configuration of the system as a function of time. It gives the condition of all the important parts of the system and the relationships between them.

Consider the simple example of a fixed mass of a known gas. The state of this system is entirely specified by gas pressure and volume. All other information about the system can be derived from these quantities using the gas equation of state.

A turbo-generator is also a system because it comprises a number of connected parts. However, the turbo-generator system is very large and complex, and many variables are needed to define its state.

Each part of the turbo-generator effects the whole in some way. Therefore, the state variables must give important information about each part of the system.

Further variables are required to give information about important relationships between parts of the turbo-generator.

For example, the state of balance of the turbo-generator is an important indication of its condition. In order to define the state of balance of the turbo-generator, it is necessary to know the state of balance of each steam turbine, generator and exciter. It is also necessary to know the relative alignment of each to its neighbours.

In the proceeding discussion, the system state will be described as consisting of a number of elements. Each state element can be specified by the values of a number state variables. For example, 'the relative alignment of two adjacent steam turbines' could be considered as a turbo-generator state element. Relative alignment could be specified by four variables, giving the angle between the shafts and their relative locations of their ends.

System observability:

A system is observable if and only if, all elements of its state can be determined on the basis of observations of the system. The nature of the observations is not intrinsically important, but practical considerations may influence the way that observations are made.

For example, the state of a system that consists of a fixed mass of gas, can be deduced from observations of its temperature and pressure, or of its temperature and volume, or of its pressure and volume. The exact mechanisms by which the three quantities are measured is theoretically unimportant.

Consider a further example. The amplitudes of a turbo-generator's vibrations are an important indicator of its condition.

The amplitudes of the vibrations would normally be measured using electrical transducers at a number of points along the machine. The electrical transducers convert the machine's vibrations into minute electrical voltage fluctuations which are represented graphically for analysis.

However, it may be equally valid for an operator to guess the amplitude of vibrations by listening to the sound of the machine.

State controllability:

A system is controllable if and only if each element of its state can be controlled. There is no intrinsic importance in the exact mechanisms by which state elements are controlled.

For example, the temperature of a fixed mass of gas may be controlled by either internal or external heating.

The state of balance of a turbo-generator can be controlled by modifying the distribution of mass along its rotating central shaft, or by adjusting the alignment of the central shafts of the individual turbines.

A concept of **ideal** state will now be introduced as it is central to any monitoring strategy. The following definition of **ideal** state only applies to man made systems, that are designed for a specific purpose.

Ideal state:

A system is in an ideal state if it capable of performing the function it was designed for.

For example, a turbo-generator is in its ideal state when it can generate electrical power according to its design specification.

In order to incorporate the above systems concepts into machine monitoring, it is necessary to introduce a concept of system **utility**.

System utility:

Utility is a function which maps 'system state' to the real line, and is intuitively conceived as a profit function, but incorporates a term to quantify injury or death of personnel.

In a given time span, the utility of a system will depend on the system's state during that time.

The utility of a system on the time interval [t1,t2] may be denoted by U:

U = utility(S,[t1,t2]),

where S represents the state of the system as a function of time.

Utility 'U' is calculated as the difference between revenue 'R' and expenditure 'E' + cost of human injury (or death) 'I':

U(S,[t1,t2]) = R(S,[t1,t2]) - (E(S,[t1,t2]) + I(S,[t1,t2]))

If 'revenue' is greater than 'expenditure' plus 'injury' then there is a net profit. If 'revenue' is less than 'expenditure' plus 'injury' then there is a net loss.

The objective of state monitoring is to maximise the total profit that derives from the system. This is equivalent to maximising the utility function:

U(S,[t1,t2])

such that:

t1 = the present time,
t2 = (nominally) the end of time.

The utility of a system can be negative, positive or zero.

The above utility is in effect an 'expected utility' because the system state over [t1,t2], where t1 is the present time, is partially unknown.

For example, suppose the state of a turbo-generator is such that it develops a serious fault at time t1 and is fully repaired by time t2. The utility of the machine will be:

$$U(S,[t1,t2]) = R - (E + I),$$

where:

1) R is the revenue from sale of electricity produced by the machine during [t1,t2],

2) E is the expenditure on the machine during [t1,t2], and

3) I is the cost associated with injury or death of personnel.

The sale of electricity will have been reduced by machine down-time whilst the machine was being repaired, and possibly by reduced generation capacity prior to repair.

The expenditure will comprise the cost of fuel to produce steam to drive the turbo-generator and the costs of repairing the machine. To be strictly accurate, monitoring costs and depreciation of the machine should also be added.

The cost associated with death or injury is very difficult to quantify because it must take into account human suffering which is purely subjective. To overcome this problem, rules of thumb are sometimes applied in which, for example, a human life may be assigned a nominal value. This seems rather unsatisfactory, but is effectively the approach of the courts, when they assign compensation to 'victims' and their relatives.

A further problem of quantifying the cost of injury, is that injury to personnel may cause loss of confidence and respect in the industry, which may have an adverse effect on business.

A major difficulty in machine monitoring is calculating the future state of the system, and therefore calculating the utility function. The monitoring problem is to maximise the utility:

U(S,[now, end of time]),

where S can be known at most on the interval:

[beginning of time ,now].

In reality, S will be only partially known.

Therefore, to properly monitor the turbo-generator it is necessary to predict its future state S. This is a considerable problem as state prediction is inherently complex even when the past state is known accurately. In fact, there will always be considerable uncertainty about past machine behaviour and therefore the machine's past state.

A Simple Example Of State Monitoring

As an example, consider the process of state monitoring for a motor car.

A car may be regarded as a system whose function is known (at least) to its owner. Suppose that it is used purely as a means of transport.

The utility of the car is measured by how well the car transports its occupants from A to B. Safety and reliability are implicit requirements.

The driver observes the car during operation, and infers its general state. If this indicates a significant lack of reliability and therefore utility, some remedial action may be required.

Suppose the engine turns over well, but takes a long time to start. Remedial options include:

1) adjust the ignition timing,

2) adjust the carburettor,

3) reset spark plug gaps,

4) seal the ignition system to exclude moisture, and

5) having the car serviced.

Remedial actions are chosen on the basis of their associated costs, and likelihoods of success. The aim is to choose remedial actions that give the best expected increase in system utility. In this case, the best improvement in starting performance.

State monitoring is likewise applied to the turbo-generator.

The approach here is essentially the same, but is complicated by greater size and complexity of the system and very high magnitudes of utility. These give rise to considerable uncertainties in expert knowledge, data, and costs of remedial action.

An approach to reasoning under these uncertainties is presented in detail in chapter 4. In the following, an intuitive approach will be used.

State monitoring prerequisites:

The following list shows the major ingredients of state monitoring:

1) knowledge of the system's ideal state,

2) the ability to make observations of the system,

3) knowledge of the relationship between system observations and system state,

4) the ability to identify significant deviations of system state from its ideal,

5) knowledge of investigative procedures, and measures for eliminating undesirable states.

Difficulties are increasingly encountered in each of these areas as system complexity increases.

The following, expands on the system theory introduced earlier.

The state of a system.

Any system state may be approximated by a finite number of real variables x1,....xn ,forming a state vector X. If the system state is **ideal** then X = [x1,x2,...,xn] = [a1,a2,...,an] = A, say. At any time, the system state will be deviant from its ideal unless X = A, i.e. unless for all i: xi = ai.

Observations of a system's state.

Generally, a system's state vector X will not be directly observable, but will be observable in part at least, via a vector of symptoms S = [s1,s2,...,sm]. During normal operation, the symptoms S will not be directly observable, but may be obtained through investigation. For example, a car's oil level is indirectly observable via the dip stick.

The symptoms S, which correspond the system's ideal state are given by S = B:

S = B when X = A.

Changes in S are a possible indication of changes in X. Therefore, deviations of S from B are a possible indication of a deviation of X from A.

Changes in the set of symptoms S, are said to be symptomatic of a change in the state X. However, changes in the symptoms are not proof of a state change because machine behaviour is also determined by operating conditions.

A simple monitoring strategy.

Assuming the current machine state to be ideal, a simple monitoring strategy is as follows.

When there is a significant change in system behaviour it is assumed that one of the following has occurred:

1) a change in the machine's state, and/or
2) a change in the machine's operating conditions.

All available data is considered and if necessary investigation performed to determine whether there has been a change in the machine's state.

If the machine's state has changed, and is no longer ideal, then remedial action must be taken to minimise the cost of the state change. This is equivalent to maximising the state utility.

Practical considerations of state monitoring.

As explained earlier, the aim of state monitoring is to maximise a utility function:

U(S,[t1,t2])

where:

S is the system state as a function of time, and [t1,t2] is the interval on which the maximisation is to be performed.

In the following t1 will represent the present time and t2 will represent a distant future time.

A practical approach to maximising the (expected) utility over [t1,t2] is as follows:

1) Maximise the utility over [t1, t2 + delta_t] where delta_t is chosen to be sufficiently small that predictions of the system state during [t2, t2 + delta_t] are likely to be accurate.

2) Collect additional data during the time increment delta_t so that the best remedial action can be chosen at t2 + delta_t. This data can also be used to better estimate the state over the next time interval.

3) Choose and implement remedial actions to be taken at time t2 + delta_t so as to maximise the system utility over the time increment.

4) repeat the above three steps for a new time increment.

Remedial decisions are usually based on evidence which derives from extensive experience of past and present machine behaviour.

In assessing certain machine behaviour it is necessary to know such things as duration, rate of change, and relation to operating conditions. Reasoning without this information, may result in incorrect diagnosis and remedial strategy.

Underlying factors that effect system behaviour.

System behaviour results from a combination of two factors:

1) system state, and
2) system operating conditions.

For example, the overheating of a car engine may result either from the engine being defective, (eg. worn) or from its being subject to unreasonable load.

An important function of state monitoring is to distinguish between changes in system behaviour associated with its state and those associated with its operating conditions.

Example of turbo-generator state monitoring:

Suppose that a turbo-generator has been operating satisfactorily for a long period of time and then shows a sudden change in behaviour.

The first question is whether the sudden behaviour change results from a change on the machine or from a change in the machine's operating conditions.

If operating conditions can be ruled out, then the change must be due to a machine state change. If operating conditions cannot be ruled out then the worst must be assumed (ie. that there has been a state change and a potential problem has therefore arisen).

The potential costs of catastrophic machine failure are so high that the state change is treated with grave concern. For example, the sudden or 'step change' in machine behaviour could result from a disintegrating rotor.

The nature of past observations gives rise to hypotheses. These are tested against all available relevant data and against any data that is obtained from investigation.

The most promising hypotheses are used as a basis for choosing remedial actions, which may include further investigations and observation.

Notes on condition monitoring.

Observations of system during its normal operation may not be sufficient to determine some elements of its state. A system's operating conditions will often be intentionally varied in attempts to evoke behaviour that will enable the system's state to be determined more accurately. This process is generally referred to as investigation.

Other processes may involve maintenance work, for example, to reveal otherwise inaccessible parts of the machine. These are sometimes referred to as investigation but are better regarded as being remedial actions. Indeed, even trivial investigation, may be regarded in the same light as remedial action. This concept of remedial action will be expanded in chapter 4 on knowledge representation and inference.

Fault conditions are expressed in terms of system states whilst symptoms are expressed in terms of system behaviour. The ability to determine fault conditions on the basis of symptoms, and so to choose successful remedies, constitutes expertise. This is obtained either through experience (case histories of system behaviour corresponding to known faults) or through system modelling.

Many real systems exhibit changes in behaviour arising from causes other than fault conditions, or transient factors. Many systems are subject to internal state changes due to wear, aging etc. It is hoped that the nature of these changes will make them easy to identify and hence easy to distinguish from changes associated with faults.

The turbo-generator as part of a larger system:

In the chapter so far, turbo-generator state monitoring has been described as the process of maximising the turbo-generator state utility.

This is a somewhat simplistic view of turbo-generator state monitoring as it does not take into account the relationship between the turbo-generator and the rest of the world.

The turbo-generator should really be considered as part of a much larger system which will be referred to as the 'power generation system'. This comprises power stations, turbo-generators, distribution networks, transformers, etc. Indeed the larger system includes everything owned by the Central Electricity Generating Board or CEGB.

The real aim of state monitoring is to maximise the total profits of the CEGB. Turbo-generator state monitoring is just an aspect of this. Therefore, the aim is to monitor each turbo-generator so as to maximise the total profits of the CEGB.

However, the power generation system is so large that state monitoring must be modular.

The function of the 'power generation system' is to supply electricity wherever it is required in the UK.

For the 'power generation system' to best perform its function, the turbo-generators must fulfil a certain role. This role of the 'power generation system' accurately defines the function of each turbo-generator depending on its type and location.

The functions of the power generation system.

The principal functions of the power generation system are as follows:

base power: to provide continuously, a minimum or **base** level of power,

top-up power: to provide the **top-up** power necessary to meet demands which exceed the **base power** levels, and which fluctuate widely,

power economy: not to produce undue excesses of power, ie. not to waste natural energy resources such as coal, oil, gas, and uranium,

safety: not to cause or give rise to injury of the population or damage to itself or other property.

The nation as a whole has a varying electrical power requirement and the purpose of the power generation system is to meet this requirement.

Throughout the course of a typical day the magnitude of the nation's electrical energy requirement will vary between minimum and maximum levels.

The minimum level is referred to as the **base power** level and must be supplied continuously. In general there will be a requirement exceeding the **base power** level, which will be referred to as the **top-up power**:

base power + top-up power = total power requirement.

The **top-up power** required will vary between zero and a maximum **top-up level**, but within those levels will vary widely in magnitude and rate of change of magnitude.

The processes of the power generation system.

In its provision of electrical power for the nation, the power generation system is involved in the processes of generating and distributing electrical power. Since the ultimate concern of this discussion is the function of the turbo-generator, only the electrical power generation (in which the turbo-generator is involved) is of interest.

The process of so called generation is really one of energy conversion, of the energy in fossil and nuclear fuels to electrical energy, (power is proportional to the rate of energy conversion).

The conversion from fuel energy to electrical energy is accomplished in two stages. Firstly the fuel is burned in furnaces and reactors to heat water and so produce superheated steam. Secondly, the superheated steam is fed into turbo-generators which convert the kinetic energy of the steam into electrical energy.

So far as the turbo-generator is concerned, the source of energy from which its superheated steam is generated is largely unimportant. The main difference between fuels with regards to energy production, is one of economy (as will be explained later).

The only major difference between nuclear and coal or oil fired power stations is that nuclear reactors are best shut down rapidly. This can mean that the turbo-generators powered by nuclear power can be expected to be deloaded rapidly, and must therefore robust in this respect.

The continuous conversion of energy.

The power generation system has very limited electrical capacitance in the sense of being able to store excesses of electrical energy which have been converted from fossil and nuclear fuels. Excess energy conversion therefore leads to fuel wastage and reduction in profits.

As a consequence of the power generation system's lack of electrical capacitance, it must try to continuously meet the nation's exact electrical power requirements.

The functions of the turbo-generator.

The broad function or role of the turbo-generator, is to convert superheated steam energy to electrical energy.

The **base power** and **power economy** functions of the power generation dictate that the most economic turbo-generators should be used to meet the **base power** requirement.

Turbo-generators in this category will be referred to as the **base power units**.

In order to fulfil their function, the **base power units** should be capable of running continuously at their maximum conversion rates, which corresponds to their maximum efficiencies.

In view of the power generation system's lack of electrical capacitance, the **top-up power** and **power economy** functions of the CEGB dictate that the nations **top-up requirement** should be met by a minimum number of **top-up power units**. Since the **top-up requirement** varies widely in both magnitude and rate of change, **top-up power units** may frequently need to vary their conversion rates, and to be switched on and off (run-up and run-down) as required.

The **safety** function of the power generation system is an overriding one, which applies to each individual turbo-generator as it does to the overall power generation system. This function plays little part in the day to day running of the machine, but comes into consideration in unusual or unexplained circumstances. For example, where there is the possibility of a potentially catastrophic fault.

Depending on their nature, faults may result in costly damage to the machine or even result in its destruction and to loss of life. Normal economic considerations may be insignificant in comparison with potential or expected loss associated with a potentially catastrophic fault, and remedial action may be required.

UNIVERSITY
OF B
LIB
ENG

Summary of turbo-generator general functions.

In general, a turbo-generator should:

1) be capable of operating at it's maximum potential load for periods of unspecified length (and continuously if it is a **base power unit**),

2) be capable of being run-up and run-down, and possibly of being rapidly and frequently loaded and deloaded (see explanatory notes below on the functions of turbo-generators for definitions of run-up, run-down, loading and deloading),

3) not to cause injury of the population or other damage (including damage to itself).

Explanatory notes about the functions of turbo-generators

1a) A **run-up** is the process of bringing a turbo-generator from a state of rest to its normal operating frequency of 50 Hz.

When a turbo-generator has been run-up, its consumption of steam energy is just sufficient to overcome energy dissipation due to frictional and other resistances, but does not enable it to generate electrical energy.

1b) A **run**-down is the process of bringing a turbo-generator from its operating frequency (50 Hz) to a state of rest.

When a machine has been run down it will normally be rotated very slowly until it has cooled down to a certain temperature. This is to prevent the machine's shaft from becoming distorted whilst hot, under its own weight

2a) **Loading** is the process of bringing a machine from a state in which it is running at its normal speed without generating electrical power, to one in which it is running at its normal speed and generating electrical power.

2b) **Deloading** is the reverse of **loading**.

3) The so called **load** on a machine is the **electrical power** which it is being drawn from it, ie. which it is generating.

2.2 Details of the physical problem.

2.2.1 Physical details of the turbo-generator.

The following contains a brief functional and structural description of the turbo-generator.

The most illuminating functional description of the turbo-generator, is that of its being:

'a means of converting the energy of superheated steam to electrical potential energy'.

Energy conversion is effected in two stages: firstly from microscopic kinetic energy of steam to macroscopic kinetic energy (of the machine's rotation) and secondly, from macroscopic kinetic energy to electrical potential energy.

The first stage in the process is accomplished by a set of steam turbines, and the second stage by a generator and an exciter (forming an alternator).

Coupling of the two conversion processes is via a physical connection in the form of a central shaft, comprising the central shafts of the turbines, generator and exciter. The components are all arranged in series (ie. in a straight line) as shown in figure 1: "Schematic diagram of a turbo-generator".

The whole arrangement (turbines + generator + exciter), is supported by either steel or concrete foundations. Each individual component has a separate attachment, and comprises both stationary and rotating parts. The stationary part is attached directly to the foundations.

The component's rotating parts are attached to the component's central shaft. This shaft is supported by a bearing at each end. Each bearing is mounted in a pedestal which is attached to the foundations. Thus each end of a components rotating part is

located in a bearing, which is mounted in a pedestal, which is attached to the foundations.

A correct relationship between the stationary and rotating parts of a component is maintained by their relative attachment to the foundations. Some designs of turbo-generator have added features, known as 'palms' which locate the stationary part of each component to its corresponding pedestals. This ensures a correct relationship to rotating parts.

Remaining to be described are: the different components;, the nature of their coupling one to another (via their central shafts and foundations); and the bearings.

The function of the turbines (described above), is to convert microscopic kinetic energy to electrical potential energy. The conversion process is achieved by forcing high pressure steam to expand across the turbine blades thus dissipating its energy. Rotation results from forces exerted by the steam on the turbine blades (each blade being attached to the central shaft).

Incidence of steam on the movable turbine blades is controlled by an enveloping duct and a series of stators (stationary blades). All this is surrounded by lagging (or housing).

The steam turbines are classified as high, intermediate, or low (1,2,3) pressure. The functional and structural descriptions above apply equally to each of the five turbines, since their differences are only related to the volume flow rate and state (pressure, temperature, etc.) of the steam which is fed into them.

The generator and exciter are similar to the turbines since they have stationary and rotating parts. Together the generator and exciter are responsible for converting macroscopic kinetic energy into electrical potential energy.

The exciters's function is to supply direct current, to energize the generator's electro-magnet. The electro-magnet is rotated by the machine's central shaft, thus creating a rotating magnetic field. This induces an electrical potential in the generator's fixed copper windings.

The intended coupling of components is via the attachment of their central shafts. Other coupling between components is incidental to their respective purposes but nevertheless, does occur via pedestals and foundations. The combined coupling effect of the central shaft attachment, mutual pedestals, and mutual foundations, gives rise to behaviour of a complexity beyond that of the individual components. Thus the behaviour of an individual component is not necessarily a result of its own processes but may be the result of the coupled behaviour of itself and other components.

The described behavioural coupling is especially important where vibrations are concerned. Vibrations emanating from one area of the machine will usually be transmitted to other parts. This has both desirable and undesirable consequences. They cause ambiguity, but enable corroboration. They also, aid identification of some faults.

Turbo-generators have journal type bearings, see figure 2: "Schematic diagram of a journal bearing". These are blocks of metal with apertures of circular cross-section. The aperture diameter is slightly larger than the (rotating) shaft diameter, leaving room for a thin oil film between the journal and shaft. The oil film or cushion forms a bearing with very low coefficient of friction. This enables smooth relative motion of the parts.

2.2.2 Details of the data used in state monitoring.

Turbo-generator state monitoring is based on the following:

1) historical data,

2) expert knowledge, and

3) process models.

Expertise and process models combined, are used interpret historical data and select remedial actions. Historical data comprises all recorded machine behaviour and associated operating conditions.

Historical data comprises:

1) machine behaviour (vibrational, etc.),

2) machine operating conditions, and

3) maintenance work.

Machine behaviour comprises all recorded observations of the machine. Observations are divided into two groups: vibrational and non-vibrational, the most important being vibrational.

At convenient times whilst running normally, a machines vibrations are measured. The measurements are then stored for future reference in case of problems with the machine, in which case the measurements can be used to assist with diagnosis.

At the outset of a diagnosis, when an abnormality is first suspected, there will be a limited set of relevant historical data. On the basis of this data some decision must be made regarding future actions. If the data yields inconclusive decisions then further data will probably be required. The existing data and resulting diagnosis may indicate which gaps in the data are most significant, and therefore which data needs to be collected.

More detailed and extensive observations of the machine are made during a diagnosis. The additional observations supplement the past observations to give a more accurate and up to date picture of current machine behaviour.

2.2.3 External conditions associated with the machine.

For any machine state change there will be a change in machine behaviour.

Given some observations of machine behaviour, it is necessary to determine any correlations with operating conditions. This can only be done, given the nature and timing of the operating conditions.

The correlation or lack of correlation between observations of machine behaviour and operating condition changes will determine whether or not the behavioural changes are due to operating conditions.

If operating conditions are the cause of the behaviour change then the monitoring task is completed and the machine is not faulty.

If operating conditions are not the cause of the behaviour change then there must have been a machine state change.

In order to determine the nature of changes in machine state it is important to filter out the effects of any exceptional operating conditions on the machine's behaviour. This will then leave only the observations of machine behaviour that are directly related to machine state changes.

It is especially important to filter out changes in machine behaviour that are due to the normal effects of wear, for example, on machine parts. Over long periods of time a machine's response characteristics can be expected to change due to normal wear and changes in alignment of components.

These sorts of changes can usually be recognised by their time-scales and progressive natures.

However, information about the response of the machine to certain operation conditions is invaluable to fault diagnosis, and must not be overlooked. For example, the response of the turbo-generator to load is an important symptom in assessing the likelihood of an electrical short in the generator's windings, which is known as 'a shorted turn'.

A 'shorted turn' occurs where electrical insulation has failed. This effectively reduces the overall resistance of part of the generator's windings which then have different thermal response to electrical load. This can result in a load dependent thermal bend in the machine which exhibits itself as a change in machine balance whenever there is a change in electrical load.

2.2.4 Machine Maintenance work.

Machine maintenance work amounts to a deliberate alteration of the machine's state, which is performed while the machine is shut down. Maintenance work will inevitably change the state of a machine and therefore its response characteristics and behaviour.

It is important to consider machine maintenance when trying to interpret changes in machine behaviour. To avoid future complications, machine behaviour is monitored closely following a machine's return to service after maintenance work. The reading that are taken can be as a datum in future monitoring.

2.2.5 Measurement of machine behaviour.

Vibrational measurements.

Measurements of the machine's vibrational behaviour are obtained via transducers attached to various parts of the machine. The transducers convert the vibrational energy into electrical signals whose amplitudes are related to those of the vibrations. The amplitudes of the electrical signals are stored in a form from which they can be retrieved and analyzed. Some useful background information on the measurement and interpretation of vibrations can be found in [Shatoff, 1976; Bannister & Osborne & Jennings, 1979].

It is possible to measure the machine's vibrations in two complimentary directions (axial and transverse) at each of the machine's 14 bearings. This gives 28 possible signals in all.

Associated with a set of signals, is the machine's mode of operation, that is, the machine's operating conditions at the time the behaviour was measured. Operating conditions include, load on the machine (power being drawn from it), rotor current (electrical current flowing in generator's conductors), rotational frequency (when the machine is being run-up and run-down), etc.

If a signal were displayed in unprocessed form as a graph of amplitude against time, its use as a monitoring aid would be extremely limited. For this reason the 'raw

signal' is usually processed by means of Fourier decomposition before being displayed. Two important means of displaying a signal are:

1) the polar plot,
2) the run-up / run-down signature.

The polar plot.

The polar plot is used to display the phase and amplitude of the frequency components of a bearings vibrations, see figure 3: "An example of a polar plot". The frequency components of the vibrations are determined using fourier decomposition.

A fourier decomposition of a signal yields a frequency spectrum showing the distribution of vibrational energy with respect to frequency. Typically, the spectrum features spikes at frequencies referred to as: once-per-rev; twice-per-rev; three-per-rev; etc. These are the frequency components of the signal at one, two, three, etc times the turbo-generator's rotational frequency (50 Hz during normal operation).

A bearing's signal is displayed using a number of polar plots, each corresponding to a frequency component of the signal. Each frequency component is displayed as a point on its polar plot; indicating its mean amplitude (distance from the origin) and phase (angle between its radius and that of zero degrees).

Normally, contours corresponding to one, two, and three standard deviations of that component of the signal are drawn around the point.

In order to show the change with time of a bearing's vibrations, a number of points may be drawn on the same plot. This is useful aid to spotting faults which have a long time-scale, for example, non-axisymmetric rotor cracks.

The run-up / run-down signature.

The run-down / run-up signature is a useful way of displaying information about the machines frequency response in graphical form, see figure 4: "An example of a signature". A different graph is used for each frequency component.

A signature is the representation of the amplitudes and phases of the frequency components of a bearing's signal, versus the rotational frequency of the machine. One signal corresponding to a run-up or run-down might give rise to a number of graphs : the once-per-rev amplitude; once-per-rev phase; twice-per-rev amplitude; twice-per-rev phase; etc., Each of the graphs is referred to as a signature (because of its visual appearance). Signatures may be obtained for any number of the machine's bearings during any run-up or run-down.

Signatures are often superimposed on corresponding previous signatures (in order to show trends, say), or on corresponding datum signatures (in order to show a change).

2.3 State monitoring strategy.

The following section outlines the perceived strategy of experts in turbo-generator state monitoring. The discussion centres around two major areas of the monitoring process, namely the diagnosis and the remedy of fault conditions.

2.3.1 The diagnostic strategy.

The aim of a diagnosis is to identify both the nature and location of any faults that exist on the machine.

Examples of 'nature of fault':

loss of mass,
movement of mass,
non-axisymmetric cracking,
etc.

Examples of 'location of fault':

high pressure turbine,
intermediate pressure turbine,
generator,
etc.

The diagnosis of fault conditions using a generate and test strategy.

The expert's diagnostic strategy is best understood in the light of a generate test strategy in which hypotheses are somehow generated (or suggested), and then tested. In this respect, the diagnostic strategy is essentially scientific and makes use of corroborative as well as refutative evidence.

Hypothesis generation.

Hypothesis generation is an important part of turbo-generator fault diagnosis. Hypotheses serve as points of focus for detailed analysis of machine behaviour. They allow the expert to examine a machine's behaviour in the light of his knowledge about individual faults, thus providing him with a framework for further analysis.

In the context of diagnosis, an expert's hypothesising may be interpreted as identifying the areas of expertise that are relevant to a problem, together with an implicit intention to apply them individually.

For the purpose of this work it is useful to consider the means by which an expert on turbo-generator faults, generates hypotheses. The term 'generate' is perhaps rather a misnomer, since it is doubtful whether an expert deliberately goes about generating or creating hypotheses. More likely, on becoming aware of some of the machine's behaviour (through graphical data, word of mouth, or whatever), certain possible faults (hypotheses) will spring to mind.

The mechanisms of hypothesis generation are not clear but undoubtedly uses some of the following:

1) knowledge about:

i) the individual turbo-generators,

ii) similar turbo-generators,

iii) turbines in general,

iv) rotor-dynamics; general dynamics; physics; etc.

2) Knowledge of serious and especially, potentially catastrophic faults.

For example, hypotheses as to a machine's faults may arise from:

1) knowledge about:

i) faults which the individual turbo-generators have had before,

ii) faults which sometimes occur on similar turbo-generators (similar by: design; behaviour; size; etc.),

iii) faults which sometimes occur, on turbo-generators in general,

iv) faults which sometimes occur on: rotary machines; any machines; etc.

2) Serious faults, whose absence from the turbo-generator must be confirmed, or else remedied, because of their potential for damage or inconvenience.

Typically, something in the perceived behaviour of the machine gives rise to its association with one or more faults. These processes of association are discussed in chapter 4 on knowledge representation and inference.

Hypothesis testing.

Hypothesis testing is the process by which hypotheses are validated.

There are essentially two approaches to hypothesis testing:

1) hypothesis refutation, and
2) hypothesis confirmation.

The refutation approach is to compare each hypothesis with existing evidence and determine any inconsistencies.

The plausibility of a hypothesis is reduced according to its level of inconsistency with existing evidence.

Hypotheses with the highest levels of plausibility are assumed to be most promising.

The hypothesis confirmation approach is to consider all evidence and deduce an overall level of confirmation for each hypothesis.

Hypotheses which have the highest levels of confirmation are assumed to be the most promising.

Either refutation or confirmation testing can be used in isolation, or the two approaches combined.

The method of refutation is safer than confirmation in the sense that every hypothesis is regarded with suspicion. There is an implicit notion that each hypothesis is just an idea or a guess. This contrasts with confirmation which contains an implicit assumption that hypotheses might somehow be confirmed. Confirmation can be a powerful tool for assessing hypotheses, but if not used cautiously, can help to foster pet theories and lead to hasty conclusions.

A particularly effective approach to hypothesis testing is to perform refutation first, thereby finding a set of plausible hypotheses. Confirmation is then applied to the plausible set to determine which of the hypotheses are most promising.

The advantage of the combined approach, is that it can greatly reduce the amount of reasoning that needs to be performed. Refutation is easier to implement than confirmation and is therefore performed first. This leaves a relatively small number of plausible hypotheses on which to apply the rather more complex confirmation.

Hypothesis testing for turbo-generator faults.

Hypothesis confirmation needs to consider a large proportion of machine behaviour in order to derive significant support for any hypothesis. This contrasts with hypothesis refutation by which a hypothesis may be found implausible by virtue of a single inconsistency with machine behaviour.

For example, the plausibility of 'a change in machine balance' can be eliminated if there is no evidence of a change in the machine's vibrations at the once per rev frequency. However, it would require several pieces of evidence to confirm 'a change in machine balance'.

The above problems encountered in confirmation are compounded by difficulties in assessing the degree of support for hypotheses. Given large amounts of evidence of machine behaviour, the assessment of the support provided is complicated. This inevitably gives rise to much uncertainty about the actual level of confirmation or evidential support for hypotheses.

Experts operate in the real world, where conditions are never ideal due to incomplete of data, lack of expert knowledge, etc. However, decisions must still be made and so experts have to use all evidence at their disposal to arrive at realistic conclusions.

2.3.2 Remedial strategy.

When a machine's behaviour has been analyzed and conclusions drawn from it, the expert has to choose a set of remedial actions so as to maximise the system's utility.

When considering turbo-generator state monitoring it is useful to regard a remedy as being:

'Any action which is performed in the light of an analysis of machine behaviour, in order to maximise the utility of the machine'.

If diagnosis indicates that the machine is in good running order, no remedial action need be performed.

If diagnosis is inconclusive due to lack of evidence, then further evidence of machine behaviour must be obtained. This may involve 'experimentation' with the machine's operating conditions to determine certain symptoms and therefore give particular pieces of evidence. The new evidence can be added to existing evidence and used in a repeat diagnosis.

If the diagnosis indicates that a serious fault may exist, then the machine may have to shut down immediately or scheduled for shut down at a convenient time.

If the machine is shut down it is most important that the service engineers are told the location and nature of the fault. Otherwise the machine down time will be prolonged whilst the fault is located.

2.4 Uncertainties associated with turbo-generator state monitoring.

2.4.1 Uncertainties associated with machine behaviour.

A major difficulty in turbo-generator state monitoring is the measurement of machine behaviour. For example, although a machine's vibrations can be measured and recorded with reasonable accuracy, they are difficult to interpret. This can lead to considerable uncertainty.

Consider the problem of comparing run-up signatures taken from a turbo-generator at different times, one taken at the latest return to service after overhaul, and one taken more recently. For an example of a run-up signature see figure on page 70.

It is the simple practice, when comparing run-up signatures, to plot a graph of the differences between them against the machine's rotational frequency.

To diagnose the fault 'non-axisymmetric cracking' from the evidence of the difference graph, it is important to assess the speed range over which a 'significant' change has occurred. An experienced expert can usually assess the speed range by looking at the shape for the difference graph. Most likely, the expert will be unable to give an exact percentage, but will be forced into a FUZZY or uncertain statement (eg. between 50 and 80 percent, or about 65 percent of the speed range).

Clearly, the difficulty in assessing the speed range over which the change has occurred is related to the assessment of significance of the changes at each of the rotational frequencies (0 to 50 Hz). Statistical techniques exist, which can be used as an aid to assessing the speed range of significant changes, but these would still express uncertainty although in statistical terms (ie. with any given speed range, there would be a level of statistical significance).

The real difficulty lies is in deciding what makes a change significant. There are so many factors that can influence the behaviour of the machine, that some behavioural variability is inevitable.

As a rule, observations of a machine's behaviour are only meaningful (and therefore useful), in the context of observations of its past behaviour which enables comparisons to be made. Thus, with regard to observations of turbo-generator behaviour, it is changes rather than absolute values which are important.

It is also important to relate the timing and duration of changes in machine behaviour to operational events involving the machine.

Whatever the means of assessment, expert or statistical analyzer, any interpretation of machine behaviour is bound to lead to uncertainty.

2.4.2 Uncertainties arising from expert knowledge.

Drawing conclusions from anomalous turbo-generator behaviour, and making decisions about optimal courses of action to bring about remedies, is the province of an expert in turbo-generator state monitoring. It is expert knowledge which enables the expert to associate fault conditions with machine behaviour. This knowledge is gained by the expert through study, and through experience of turbo-generators, rotary machines, and machines in general.

Even if it is definitely known that a turbo-generator has one single fault, and the expert has every relevant piece of evidence to hand without any uncertainty, there will still be uncertainty in his conclusions. This uncertainty is especially obvious in cases where incorrect diagnoses could lead to large costs, and in which the expert is much more guarded in his assessments.

Sources of uncertainty in expert knowledge include incomplete and false or conflicting knowledge. A fault which occurs on a machine will most likely give rise to changes in machine behaviour.

Observers tend to build up mental pictures of faults in terms of machine behaviour. This happens through continued exposure to the behaviour of machines on which faults are known or found to exist.

When an observer has sufficiently complete mental pictures of possible machine faults in terms of machine behaviour, he is called an expert (but of course, there are varying degrees of expertise).

An expert's perceptions of machine faults will be flawed for the following reasons:

1) when faults occur on the machines, not all of the behaviour which they give rise to is appreciated and remembered.

2) there is sometimes false reasoning in connection with machine behaviour, in that some aspects of behaviour which are not be associated with the fault, are falsely attributed to the fault.

3) incorrect assessments are easily made about the relative importance of the different behavioural characteristics of faults (for example, the aspects of behaviour which are most easily visible or most obvious, are taken to be more important).

4) most importantly, very limited experience is available in the field of turbo-generator fault diagnosis (due to the low rate of occurrence of each fault), and so extensive uncertainty about the behaviour due to the different faults, is bound to exist.

The belief that experts' knowledge is generally uncertain, is clearly corroborated by the fact that experts often arrive at different conclusions from each other when presented with the same scenarios.

2.4.3 Incomplete knowledge of machine behaviour.

Since turbo-generators are large and complex machines they exhibit large amounts of complex behaviour, and it is unlikely that all symptoms relevant to hypothesis testing will be available.

Turbo-generators run faultlessly for most of the time, and since measurements of machine behaviour are limited during such times, past data is usually rather limited. Furthermore, not all symptoms can be easily collected without costly investigation which needs to be justified on the grounds of maximising system utility.

In view of the difficulties imposed by limited availability of observations of machine behaviour, it is to be expected that only limited evidential support can be assigned to conclusions. Any conclusions that are reached, will have uncertain support associated with them due to unavailability of some of the relevant symptoms.

Human experts are quite good at working with limited evidence or limited numbers of symptoms, and arriving at sensible conclusions albeit with some uncertainty. Some uncertainty is unavoidable because there is no means of drawing definite conclusions when the underlying evidence is uncertain.

2.4.4 The inability to quantify costs associated with fault conditions.

When the expert has diagnosed the causes of suspect machine behaviour, he has then to decide the optimal remedial strategy. In order to do this, the expert must have a good appreciation of the costs of the available remedies, and must consider the weights of evidence for and against the diagnosed faults.

Remedial decisions are effectively based upon a strategy of maximising a utility, which was described at the beginning of the chapter.

There is inevitable difficulty in quantifying evidence for and against the existence of faults, and quantifying the actual costs of faults and remedies. As costs cannot be quantified, qualitative costs are sometimes used as a basis for making decisions.

The next chapter discusses, in general terms, the types and problems of uncertainty, and methods of representing and reasoning with uncertainty.

Methods for representing uncertainty, costs and therefore utility, are discussed in chapter 4 on knowledge representation and inference. A representation is described which is designed to cope with the extensive uncertainty that arises in turbo-generator state monitoring. One problem is to find a knowledge representation which is flexible and comprehensive but also concise.

2.5 Early attempts at producing computer systems for fault diagnosis on turbo-generators.

This section outlines two attempts at producing computer systems for fault diagnosis on turbo-generators. One system was produced in Mexico [Kubiak & Rothhirsch & Aguire, 1984] and another in Japan [Kurihara & Nishikawa & Nagahashi, 1983]. Both systems are essentially procedural rather than knowledge based.

The mexican system implements a decision tree, shown in the reference as a large flow diagram, which appears quite complex, and does not take into account uncertainty. It is not clear what is supposed to happen at a decision node when a 'yes/no' answer is inappropriate. Presumably, a guess is required.

It is hard to see how this method can succeed with such a rigid approach. At the end of the reference, the authors recognise this rigidity and introduce the idea of a knowledge based approach in which they can represent typical vibration patterns of faults. But still, there is no mention of uncertainty.

The Japanese system incorporates the following features:

1) in the case of rapidly increasing vibration levels, the system can display operating guidelines,

2) the system can perform Fourier analyses of data, and help to determine faults by means of crude pattern matching based on a fault-to-symptom 'matrix',

3) data can be stored for later use,

4) the system can perform balancing calculations, based on run-up vibrations, as to the weights that need to be attached to different rotors in order to correct out-of-balances,

5) the system can display Nyquist, Spectral, and Campbell diagrams of vibration data that are obtained during run-ups and run-downs.

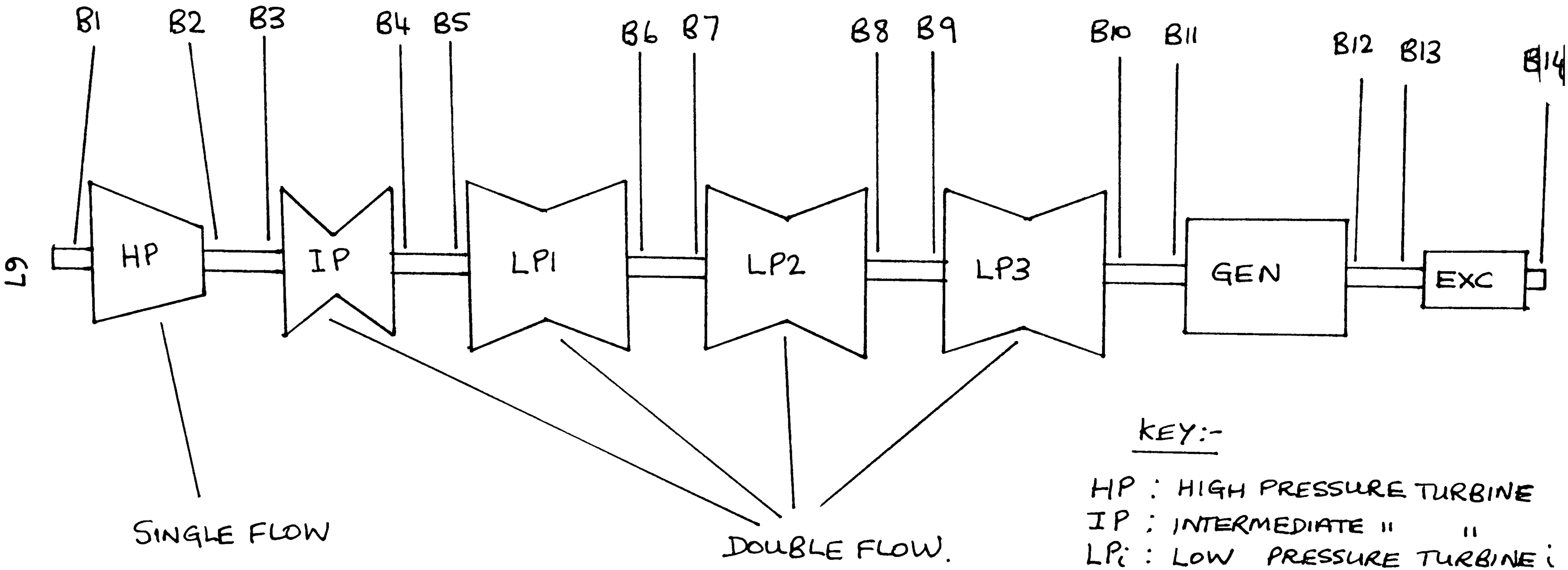
It may be significant that the Japanese paper which describes the diagnostic system shows the turbo-generator as having only three turbines and a generator. This is somewhat simpler that those used by the CEGB which have six turbines, a generator and an exciter.

On the basis of the description of how fault diagnosis is performed, it seems unlikely that this approach could be successfully applied to the CEGBs machines.

The fault-to-symptom matrix appears to be reasonably complete but the relative importance of different symptoms in fault diagnosis is not specified. Nor is there any mention of the handling of uncertain information, and presumably therefore the system uses internal cut-offs for each symptom by which it decides whether or not the symptom is present.

The strength of the system seems to be in its ability to speed up the acquisition and processing of vibration data thereby simplifying fault diagnosis for the expert, rather than fault diagnosis itself.

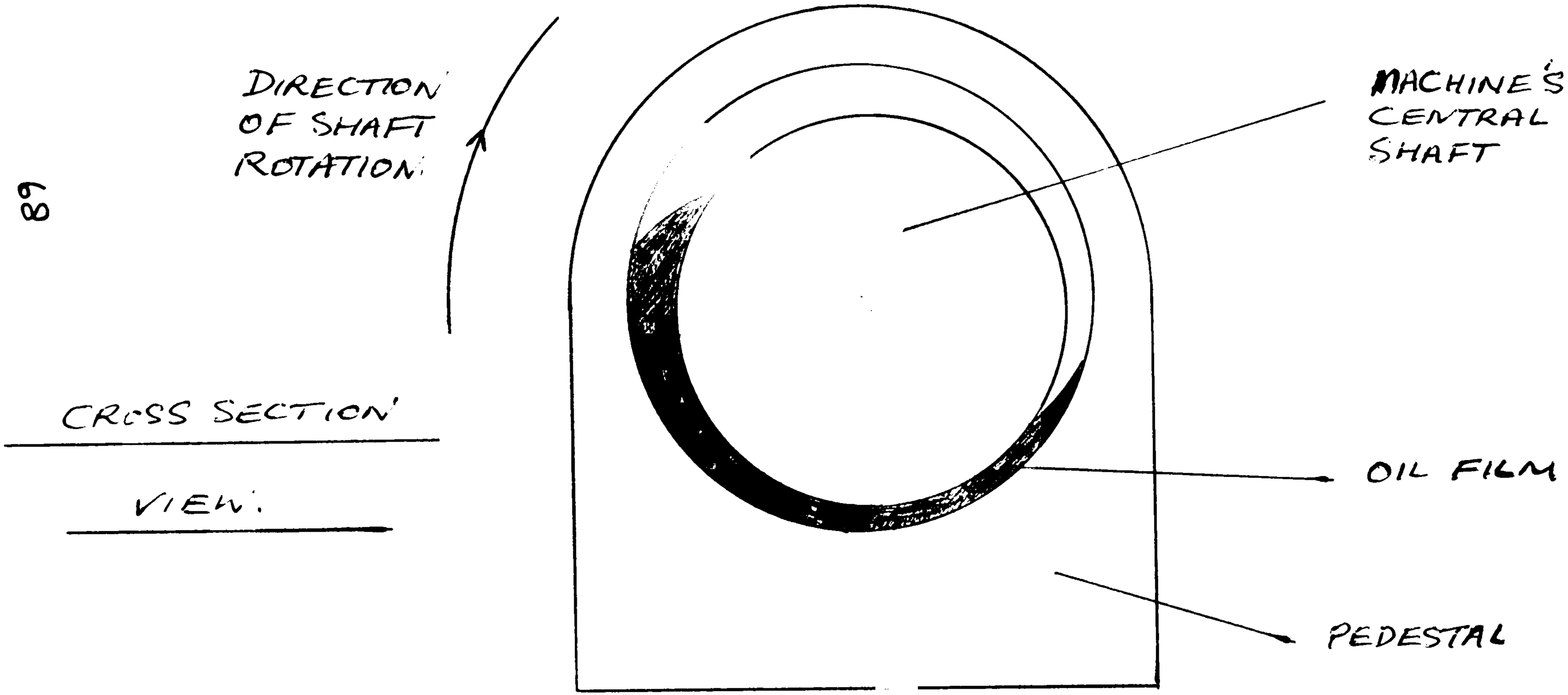
FIGURE 1 : SCHEMATIC DIAGRAM OF A TURBO-GENERATOR.



KEY:-

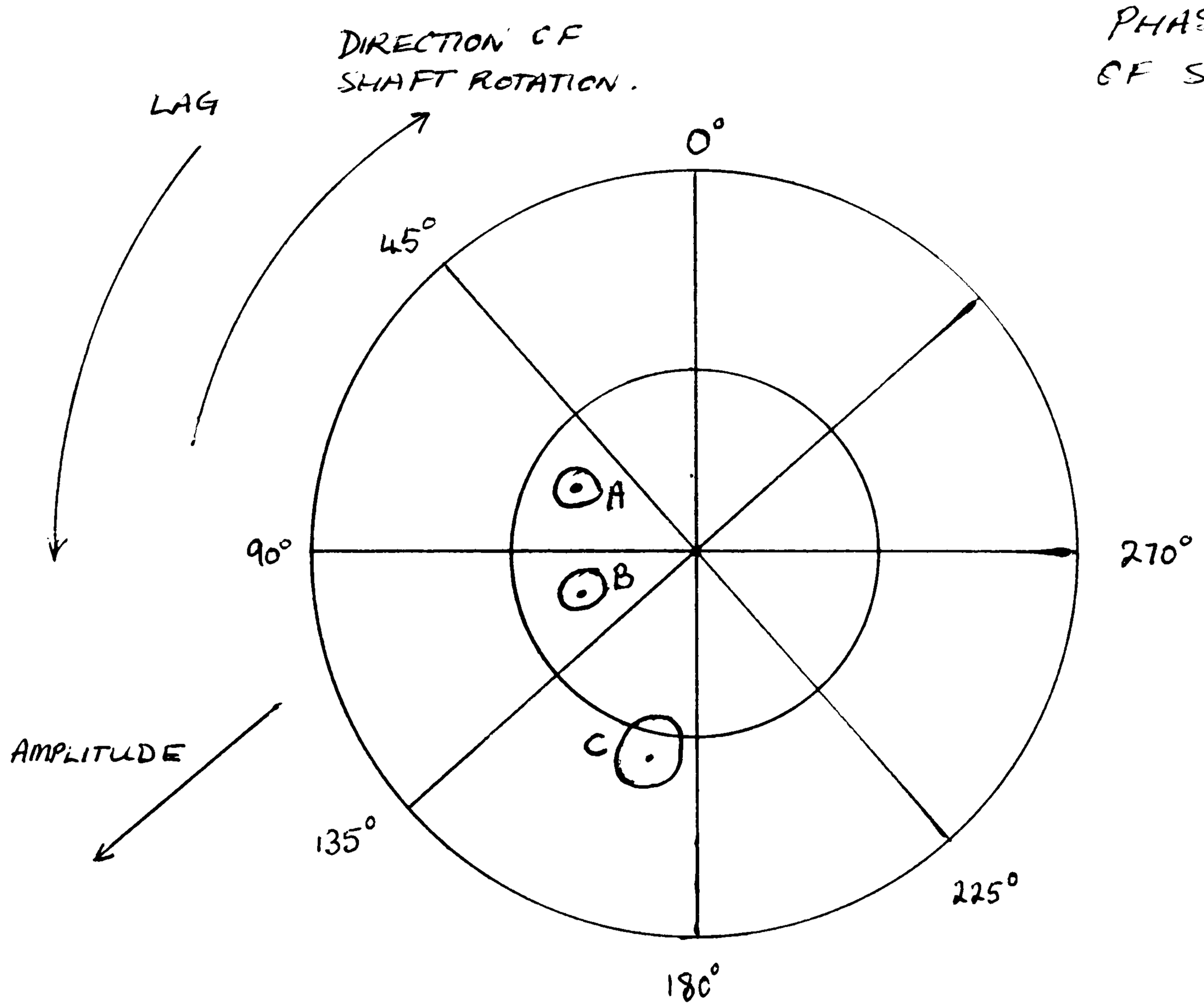
HP : HIGH PRESSURE TURBINE
IP ; INTERMEDIATE " "
LPi : LOW PRESSURE TURBINE i
GEN : GENERATOR
EXC : EXCITER
Bi : BEARING i

FIGURE2 : SCHEMATIC DIAGRAM OF A JOURNAL BEARING.



CROSS SECTION VIEW.

FIGURE 3: AN EXAMPLE OF A POLAR PLOT (OF SHAFT BALANCE)

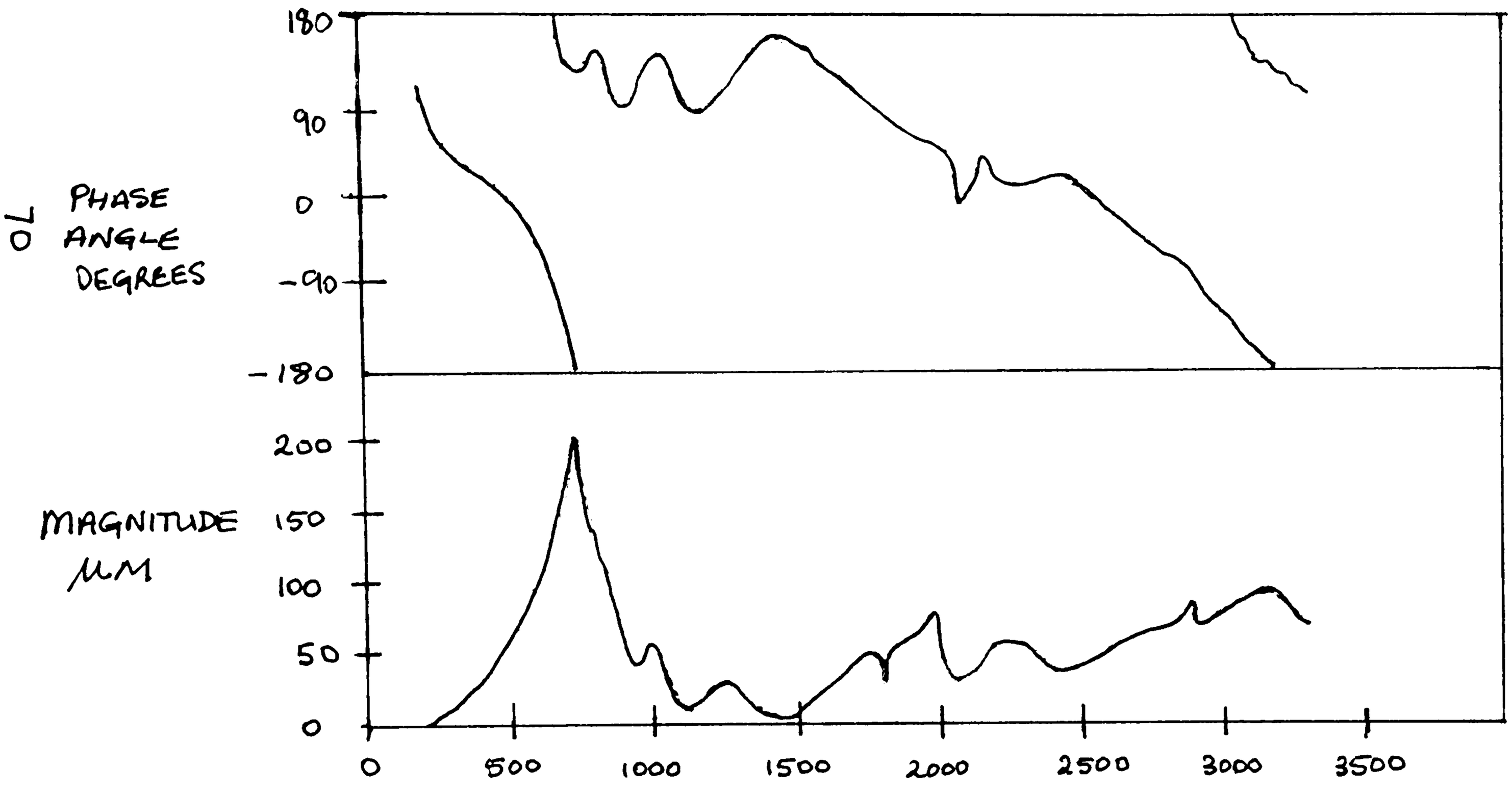


PHASE AND AMPLITUDE OF SHAFT OUT OF BALANCE.

VIEW FROM EXCITER END OF BEARING NUMBER 3 - VERTICAL DIRECTION

A JANUARY
B FEBRUARY
C MARCH

FIGURE 4 : AN EXAMPLE OF A SIGNATURE (FROM A RUN-UP)



CHAPTER 3

REASONING UNDER UNCERTAINTY

3.1 Introduction

In chapter 1, a number of models of reasoning were described, but without special reference to the treatment of uncertainty.

Chapter 2 explains in detail that a model of the reasoning for turbo-generator state monitoring must be able to cope with wide ranging uncertainty.

This chapter considers possible ways implementing an expert system for turbo-generator state monitoring, from the point of view of choosing an adequate model of reasoning under uncertainty.

Various models of uncertainty are considered, and in particular, 'Support Logic' [Baldwin, 1986, 1987] which is based on a theory of evidence.

Another model of uncertainty is also introduced. This model is tailored specifically for modelling the reasoning under uncertainty that is necessary in turbo-generator state monitoring. The model is described in detail in chapter 4.

3.1.1 An intuitive overview of the uncertainty in turbo-generator state monitoring.

Uncertainty arises in turbo-generator state monitoring for a variety of reasons which are described in chapter 2.

The uncertainty is wide ranging and comprises various types including 'probabilistic uncertainty', 'fuzziness' and 'evidential uncertainty'.

Methods of modelling these three types of uncertainty will be described in some detail, later in the chapter, but for now will be intuitively characterised as follows.

3.1.2 Probabilistic uncertainty

Probabilistic uncertainty occurs in processes which are stochastic (ie. that are governed by the laws of probability). The probability, represented by a single number, of a stochastic event is a measure of the expectation that the event will occur. For example, the probability of rolling a die and obtaining a 5 is (1/6). When an experiment is performed, the two possible and distinct outcomes are 'a 5' and 'not a 5'. There is no possibility of a compromise.

3.1.3 Fuzziness

Fuzziness is often confused with probabilistic uncertainty, but is quite different. Fuzziness does not relate to stochastic events, but instead relates to vagueness. For example, consider the following fuzzy set definition for the vague concept 'is a tall person':

Person Height	'is a tall person'
4'-0"	0.0
5'-9"	0.0
6'-0"	0.5
6'-2"	1.0
8'-0"	1.0

For each height in the range 4'-0" to 8'-0", this set associates a number in the range 0.0 to 1.0, which is conceived as the level of membership of a person of that height, to the set 'tall'. For instance, a person of height 6'-0" has a set membership level of 0.5, and a person of height 6'-1" has a set membership level of 0.75.

However, if a person is tall with set membership level 0.75, it does not mean that the probability of their being tall is 0.75.

It could be argued that there is no objective definition of the fuzzy set 'is a tall person', and that there is a stochastic element in the process of selecting a person who can then give a subjective opinion of the set membership level.

3.1.4 Evidential uncertainty

Evidence theory is essentially a extension of probability theory.

Probabilities are reinterpreted as supports, and may be specified more flexibly than in conventional probability theory, and the rules of inference are more general.

Fuzziness may also be represented as a support. For example, the fact than 'John is tall' at a level of '0.75', may give some support to the proposition that 'john is potentially a good high jumper'.

Weakly held beliefs can be assigned weak but significant support which can be incorporated in reasoning to add weight to arguments. In particular this enables the incorporation of default and heuristic information.

It is also possible for numbers of small and insignificant supports to 'combine' and give an overall significant support.

An evidential approach is more flexible in dealing with the probabilistic and fuzzy concepts that are needed to perform human-like reasoning.

'Support logic' is a flexible model of evidential reasoning that incorporates all the advantageous aspects described above, and will be described later in this chapter.

3.2 Discussion of uncertainty.

The representation of uncertainty in expert systems has been a continuing dilemma for AI researchers interested in the practical applications of knowledge representation and inference techniques.

There have been a number of proposed solutions to this dilemma including adapted Bayesian inference schemes, ad hoc confirmation measures, adaptations of the theory of fuzzy sets, and combinations of all these techniques.

In this section the following familiar and well founded numeric models of uncertainty will be examined:

1) Probability theory,

2) Shafer's evidence theory,

3) Zadeh's possibility theory.

Models of uncertainty such as non-monotonic logic are not considered here as there is a basic difference in the type of uncertainty that is intended to be represented by the numeric and non-numeric approaches. However, non-numeric models of uncertainty have already been briefly considered in the knowledge representation section of chapter 1.

The need for reasoning under uncertainty arises where knowledge is insufficient in one or more of the following ways:

1) knowledge is partial: for example, when the answers to some relevant questions are not known.

2) knowledge is not fully reliable: for example when the answers to questions are not accurate and exact.

3) the representation language is inherently imprecise: for example, when the concepts are vague and therefore open to varying interpretations.

4) knowledge deriving from multiple sources is conflicting: for example, when different experts give opposing answers to questions or when different sensors show conflicting readings.

Causal relationships between any two world events in a real world are not always easy to specify, partly because every event can seem to be related to numerous other events. Therefore, for an expert system to predict an event, it may require a rule involving a prohibitive number of antecedents.

A simple and popular solution to this problem is to have a restricted or approximate model of the world which hopes to include major interactions whilst excluding unnecessary detail. The difficulty with this approach is that of

restricting the range of applicability of the model and also in finding an approximate but acceptable accurate model.

An easier solution is to specify heuristic rules along with some measures of confidence in the relationships embodied in them. Such rules add to the uncertainty of the inferences obtained from premises which are uncertain. If the information concerning a problem is partial or approximate, then the problem can only be solved approximately, ie. with uncertainty. A solution without uncertainty may be possible when complete and exact information is available. But in this case is may still be desirable to accept an approximate solution in order to reduce computational effort.

There are essentially three aspects to the problem of reasoning with uncertain information:

1) representation of uncertain information,

2) combination of bodies of uncertain information, and

3) the drawing of inferences using uncertain information.

3.2.1 Representation of uncertain information.

A principal aim of finding a representation of uncertainty is to enable expert systems to perform reasoning under uncertainty by means of symbolic processing.

The problem then, is how to accurately represent inexact concepts and information using a formalised representation consisting of a finite number of symbols, each having a predefined and exact interpretation.

For a representation to be successful it should have a general similarity with the type of concepts being represented and should be as simple as possible. This is to enable easy and economical representation of concepts and thereby simplify knowledge acquisition, inference, and explanation of these.

In order for an expert system to make decisions it needs to consider the existing state of the world which effects the relative payoffs of different actions. It is this information, about the state of the world, which is most often uncertain.

Chandrasekaran [Chandrasekaran, 1986] argued that if extensive human interaction is part of the decision making system then it would seem logical to allow the human expert in the domain to resolve the uncertainties.

Chandrasekaran also states that humans do not use a single method for resolving uncertainties of various types and that it is pointless to attempt to find a normative uncertainty calculus which will enable understanding of intelligent human problem solving strategies.

However, it is easy to argue that for mechanised decision making systems that do not involve extensive expert interaction, a formalism is needed for representing uncertain world models.

3.2.2 Combination of bodies of uncertain information.

Combining information from different sources is an important and basic problem in the reasoning processes.

The combination of information may be considered at two levels. Firstly, there is the problem of merging knowledge at the level of individual facts, to obtain composite effects. For example, the facts: f1 and f2 could be combined logically by conjunction or disjunction.

Secondly, there is the problem of obtaining an overall perspective by combining whole bodies of knowledge.

3.2.3 Drawing inferences from uncertain information.

Assuming that the problems of representation and combination of uncertain information have been resolved, the problem then is how to solve problems by drawing inferences.

Problem solving knowledge comprises causal rules of the form:

if p *then* q

where p and q are logical expressions which describe elements of the possible states of the world.

Causal rules are worked on by the axioms of inference of the logical system. In general this is called reasoning.

Reasoning can be either:

1) inductive, or

2) deductive.

Inductive reasoning consists of evolving knowledge to explain causal links. This type of reasoning takes place in learning systems, for example.

Deduction is the main concern of this work and will be examined in the following.

Consider a causal rule linking a variable X taking values in U_X, and a variable Y taking values in U_Y, expressed by a collection of rules:

if 'X is in A_i' then 'Y is in B_i' with g_i

where A_i U_X and B_i U_Y and g_i represents the degree of uncertainty of the rule. g_i can be a probability, possibility or truth value measure, and may be only approximately known, in which case the set of possible values may be represented by a fuzzy number.

The problem is to use the uncertain body of information X and the above rules to deduce something about Y.

If the elements of U_X and U_Y are assumed to be mutually exclusive and if A_i and B_i are not singletons, then the causal link is incompletely defined. If A_i and B_i are singletons then in a sense, the description of a causal link is complete.

These rules, in conjunction with the information available about X, and in conjunction with the axioms of inference, enable new conclusions to be derived.

In classical logic the two main rules of detachment are:

Modus Ponens :

p --> q

p

---

deduce q

Modus Tollens :

p --> q

NOT q

---

deduce NOT p

If p --> q, then knowledge that q is true makes p more credible because q is a necessary condition for p. Similarly, knowledge that p is false makes q less credible. This type of reasoning is not a part of classical logic and is sometimes referred to as abductive or false reasoning. However the principal of this type of reasonable is intuitively acceptable but requires some additional information on the validity of deductions that are obtained in this way. I.e:

if p --> q then what can be deduced about p given that q is true.

3.2.4 Probability.

Probabilistic representation of uncertainty.

The basic idea of a probabilistic representation of uncertainty is to associate a probability measure with each proposition p, where p P, and P is a finite set of propositions.

A probability can be realistically interpreted in one of two ways:

1) as a degree of confirmation, or

2) as a relative frequency in the long run.

From the point of view of knowledge representation and performing inference, probability must be interpreted as the former of these two, ie. as a degree of confirmation [Cox, 1946].

Probability is used to represent logical truth as follows:

if p is a false proposition then: PR(p) = 0.0,

if p is a true proposition then: PR(p) = 1.0.

However (assuming a maximum entropy approach), a state of complete ignorance about the truth or falsity of p is represented by

PR(p) = 0.5.

But this is also the appropriate representation for the situation of equal evidence for truth and falsity.

Clearly, a more flexible representation is needed. One suggested approach is to also consider the standard deviation of the probability. In this case, a situation of complete ignorance is distinguished from a situation of balanced confirmation, by associating a large standard deviation. This is especially true if the distribution is normal.

Combining probabilities.

Consider the two propositions p1 and p2. These propositions may be combined to give a compound proposition p which is either the conjunction or the disjunction of p1 and p2. The probability PR(p) can then be calculated from the probabilities PR(p1) and PR(p2) as follows.

Conjunction of p1 and p2:

$$PR(p1 \text{ AND } p2) = PR(p1) * PR(p2|p1)$$

$$= PR(p2) * PR(p1|p2)$$

Disjunction of p1 and p2:

PR(p1 OR p2) = PR(p1) + PR(p2|NOT p1)
- PR(p1) * PR(p2|NOT p1)

Negation of p1 or p2:

PR(NOT p1) = 1.0 - PR(NOT p1)

PR(NOT p2) = 1.0 - PR(NOT p2)

The form of above expressions is quite simple. However, there are significant problems in evaluating the conditional probabilities which are of the form PR(p1|p2), PR(p2|p1), PR(p1|NOT p2), and PR(p2|NOT p1).

The situation is simplified considerably under the assumption that p1 and p2 are independent, in which case:

PR(p1|p2) = PR(p1)

PR(p2|p1) = PR(p2)

etc.

Therefore, if p1 and p2 are independent, the expressions for PR(p1 AND p2) and PR(p1 OR p2) can be simplified as follows:

PR(p1 AND p2) = PR(p1) * PR(p2),

PR(p1 OR p2) = PR(p1) + PR(p2) - PR(p1) * PR(p2).

Probabilistic inference.

Probabilistic inference works on knowledge of the form:

p --> q,

which have associated probabilities PR(p --> q).

If PR(p) is known then something can be deduced about PR(q), and conversely. However, the nature of what can be deduced depends on the precise interpretation of p --> q.

Three possible interpretations are:

1) PR(p --> q) = PR((NOT p) OR q)

2) PR(p --> q) = PR(q|p).

3) PR(p --> q) = PR(p q).

Prade [Prade, 1985] has considered the two of these and provided the following extensions.

1) Where p --> q is interpreted as (NOT p) OR q:

PR(p --> q) >= a
PR(p) >= b
---
deduce PR(q) >= MAX(0.0,a+b-1.0)

and:

PR(p --> q) >= a
PR(p) <= b
---
deduce PR(p) >= MIN(1.0,1.0-a+b).

2) Where PR(p --> q) is interpreted as p(q|p):

PR(q|p) >= a
PR(q) >= b
---
deduce PR(q) >= a * b.

and:

PR(q|p) >= a

PR(p) <= b

---

deduce PR(p) <= MIN(1.0,b/a) a/= 0.0.

From these examples, it is clear that the bounds on the inferred probabilities depend on the interpretation given to PR(p --> q).

The second interpretation is the most problematic as is relates two different event spaces, one for p and one for q,in which the probability of q is conditioned by p. In order to determine this probability PR(q|p) it is necessary to know all the conditional probabilities connecting the two event spaces. In practise these conditional probabilities are unlikely to be easily calculated.

Medical diagnosis is one problem area where the second interpretation of the probability is valid. For example, consider the problem of diagnosing illnesses q_i from symptoms p_j. The illnesses $\{q_i\}$ and the symptoms $\{p_j\}$ form different spaces.

In order to perform diagnoses it is necessary to know the conditional probabilities $PR(q_i|p_j)$ which are probabilities of the illnesses given the symptoms. These are unlikely to be readily available, but may sometimes be obtained using Bayesian statistics from probabilities of the symptoms given the illnesses.

Bayes theorem is as follows:

$$PR(s_i|p_j) = \frac{PR(p_j|s_i) * PR(s_i)}{PR(p_j)}$$

$$= \frac{PR(p_j|s_i) * PR(s_i)}{\sum^{all\ 1} PR(p_j|s_i) * PR(s_i)}$$

3.2.5 Evidence theory.

Evidence theory was started by Dempster [Dempster, 1967] and then extended by Shafer [Shafer, 1976]. Shafer-Dempster theory is a generalisation of probability theory with its roots in the theory of upper and lower probabilities. Consequently,

some of the important ideas in the theory can be thought of in terms of upper and lower probabilities.

In this section, the basic ideas of the theory are described, but without going in to much detail.

1) Let q be a variable whose set of possible values is represented by U, which is referred to as the 'frame of discernment' for q.

2) Let P(U) be the power set of U comprising all nonempty subsets A of U:

$A \subseteq U$, and $A \neq \phi$

3) Propositions are of the form:

'The value of q is in A'

where A is a subset of U, ie. $A \subseteq P(U)$.

4) The degree of support provided to a proposition A is determined by a numerical function m which is termed 'the basic probability assignment' and which represents the body of available evidence.

The function m maps elements of P(U) to a number between 0.0 and 1.0 subject to the constraints:

$m(\phi) = 0.0$

and

$\sum^{A \subseteq P(U)} m(A) = 1.0.$

Complete ignorance occurs when the whole weight of evidence is assigned to U, and is represented by an m function:

$m(U) = 1.0.$

To obtain a measure of the total belief committed to A, a belief or credibility function is defined:

$$Bel(A) = Cr(A) = \sum^{B \subseteq A} m(B).$$

A subset A of a frame U is called a focal element of the belief function Bel over U if m(A) > 0.0. The union of all such focal elements is called the core C of the belief function over U. Therefore, for any subset A of P(U), Bel(A) = 1.0, if and only if $C \subseteq A$.

A plausibility function Pr(A) is also defined, which is complimentary to the credibility function Cr(A) such that:

$$Pl(A) = 1.0 - Cr(\text{NOT } A),$$

which can be expressed in terms of the m function:

$$Pl(A) = \sum^{A \cap B \neq \phi} m(B).$$

To see how the credibility and plausibility functions work, it is useful to consider the following simple example.

Suppose that a variable q has a frame of discernment:

$$U = \{ a, b, c \}$$

which has a power set:

$$P(U) = \{ a, b, c, ab, ac, bc, abc \}.$$

All available evidence as to the value of q is specified by the m function. For example:

set	m	Cr	Pl.
a	0.2	0.2	0.5
b	0.1	0.1	0.7
c	0.1	0.1	0.5
ab	0.2	0.5	0.9
ac	0.0	0.3	0.7
bc	0.3	0.5	0.6
abc	0.1	1.0	1.0

In this case, the core C of U is:

C = { a, b, c, ab, bc, abc }.

The credibility and plausibility of {ab} is given by:

Pl(ab) = m(a) + m(b) + m(ab)

= 0.2 + 0.1 + 0.2

= 0.5

Cl(ab) = m(a) + m(b) + m(ab) + m(ac) + m(abc)

= 0.2 + 0.1 + 0.2 + 0.0 + 0.1

= 0.6

It is worth noting that although the m value for the subset {ac} is 0.0, the credibility and plausibility functions are still non-zero by virtue of the m values assigned to the subsets and supersets of {ac}.

In the special case where the focal elements of the frame U form a nested structure such that:

$$A_1 \subseteq A_2 \subseteq A_3 \ldots \subseteq A_n$$

the credibility and plausibility functions reduce to the necessity and possibility measures defined in Zadeh's possibility theory, which is described in a later section.

This represents a situation in which evidence is imprecise but consistent. With further evidence it is possible that the value of q could be determined precisely.

In contrast, if the focal elements are all pairwise disjoint then the evidence is inconsistent but precise, the probabilities being mutually exclusive. This situation is equivalent to having a probability distribution over U.

Combining bodies of evidence.

Different bodies of evidence can be combined using Dempster's rule which is a generalisation of Bayes' rule.

The credibilities and plausibilities that are due to combined bodies of evidence are most easily calculated from the combined basic probability function.

Let m_1 and m_2 be two basic probability assignments representing two uncertain bodies of evidence for the same frame of discernment. These are combined to give an overall basic probability assignment m, using Dempster's rule as follows:

$m(\phi) = 0.0$

and

for all $C \neq \phi$, $m(C) = K * \sum^{Ai \cap Bj = C} m_1(A_i) * m_2(B_j)$

where

$K^{-1.0} = \sum^{Ai \cap Bj \neq \phi} m_1(A_i) * m_2(B_j)$

The method is most easily understood using a geometric picture provided by Shafer [Shafer, 76].

The basic probability mass function m_1 is depicted as partitioned unit line segment comprising sub-segments such that the length of each sub-segments corresponds to the mass attributed to one of its focal elements. To combine the effects of the two mass functions, consider a unit square with the two sides representing m_1 and m_2 respectively. Then the intersection of any two strips, say A_i and B_j commits exactly $m_1(A_i)*m_2(B_j)$.

Some subset C of U may have more than one rectangle committed exactly to it, which accounts for the summation in the above formula.

K^{-1} represents the portion of mass that is committed to the empty set ϕ. This mass is redistributed over the m function by multiplying the area of each rectangle by K.

In practise, renormalisation can give rise to counter-intuitive results when combining evidence from highly dissonant sources. Dempster's rule brings out the items of concurrence between the two sources by redistributing unattributed evidence. However, the final function gives no indication of the conflict between the two sources. For example, if the frame of discernment of a variable q consists of {a,b,c} and two sources give:

$m_1(a) = 0.99$ $m_1(b) = 0.01$ $m_1(c) = 0.00$

$m_2(a) = 0.00$ $m_2(b) = 0.01$ $m_2(c) = 0.99$

the combination is given by:

$m(a) = 0.00$ $m(b) = 1.00$ $m(c) = 0.00.$

Although the model is fairly flexible in respect of being commutative and associative, its use is limited to combining evidence relating to identical frames. For example, evidence sources with frames of discernment {a,b,c} and {b,c,d} cannot be combined using Dempster's rule.

Inference

A major drawback with the Shafer-Dempster model of uncertainty is the absence of effective decision making procedures. Ignorance and uncertainty are directly represented in the belief functions and must be carried through any system of combination or inferencing.

The problem of inferencing in this context can be stated as follows. Given a set of causal rules of the form:

'if p_i then q_i'

where p_i and q_i may possibly belong to different frames of discernment, and a body of uncertain evidence for p_i's, deduce what can be said about q_i's frame of discernment.

A set of inference rules for the case where p_i and q_i have the same frame, are given in [Garvey & Lowrance & Fischler, 1981). These rules show how limits for the plausibilities and credibilities of q_i can be derived from those of p_i.

For propositions corresponding to different frames, inference becomes much more difficult because all the rules connecting the two frames must be considered.

3.2.6 Possibility theory.

Possibility theory was first proposed by Zadeh as a development of fuzzy set theory, for representing vagueness in linguistic terms.

For a given frame of discernment, a possibility distribution may be defined in a similar way to a probability distribution. However, a possibility distribution is intuitively different from a probability distribution. This is clearly illustrated by the following example given by Zadeh.

Consider the statement:

'Hans ate X eggs for breakfast',

where X can take values in U = { 1, 2, 3, 4, ... }.

A possibility distribution P(u) may be associated with X, by interpreting P(u) as the ease with which Hans eats u eggs. A probability distribution PR(u) may be similarly defined, which gives the probability that Hans ate u eggs for breakfast. The two distributions might compare as follows:

u	1	2	3	4	5	6	7	8
P(u)	1	1	1	1	0.8	0.6	0.4	0.2
PR(u)	0.1	0.8	0.1	0	0	0	0	0

This shows that a high possibility does not imply a high probability because the fact that an event is possible, does not make it likely. However, a low possibility does imply a low probability because an impossible event must also be improbable.

There is a continuing debate about whether a fuzzy or possibilistic representation of uncertainty is really necessary, or whether probability should be used exclusively.

Cox [Cox, 1979] has proved that if the 'plausibility' of a proposition is to be represented by a real number and the resulting calculus is to be consistent, then the 'axioms' of probability follow logically. This means that fuzzy logic, which is not identical to probability, must necessarily violate the basic consistency requirements.

From this, some would conclude that probability is the only calculus for representing and reasoning about uncertainty. Others avoid this argument by claiming that 'vagueness' is essentially different from probabilistic uncertainty and so needs a different formalism. Then again, it could be argued that vagueness is uncertainty about intended meaning, and can also therefore be represented by a probability distribution.

Another justification for the fuzzy approach is given by Zadeh [Zadeh, 1986] who states that probability is an inadequate representation of uncertainty because of its lack of expressiveness, especially for describing fuzzy sets and fuzzy probabilities. For example, how can it be represented that:

'It is very likely that Mary is young',

in which 'likely' is a fuzzy probability and 'young' is a fuzzy predicate. Furthermore, how can it be inferred from the proposition, an answer to a question such as:

'What is the likelihood that Mary is not very young ?'

Zadeh argues that problems like this cannot be solved using conventional probability based methods, but can be dealt with effectively by use of fuzzy sets.

<u>The basic formalism of possibility theory.</u>

The concept of a probability distribution as defined by Zadeh [Zadeh, 1978] as follows.

Let F be a fuzzy subset of a universe of discourse U, characterised by the membership function M_F. The grade of membership $M_F(u)$ may be interpreted as the compatibility of u with the concept labelled F.

Let X be a variable taking values in U. Then, F acts as a fuzzy restriction R(X), on the values X can take.

The proposition ' X is F ', translates to R(X) = F.

This associates a possibility distribution P_X with X, which is postulated to be equal to R(X):

$P_X = R(X)$,

and the possibility distribution function p_x is numerically equal to M_F:

$p_x = M_F$.

If U is the exhaustive set of possible values of X, then at least one of these values must be completely possible. In this case, the possibility distribution is said to be normalised.

Possibility theory has been built on the theory of fuzzy sets and is best suited to representing vague linguistic predicates like: tall, old, dark, fast, etc.

Also, by virtue of the cardinality function of a fuzzy set, concepts such as: some, most, few, etc, can be represented. This is discussed in detail by Dubois and Prade [Dubois & Prade, 1985].

Possibility distributions are not well suited to representing uncertain information such as the approximate height, age, colour, etc. of a person.

Combining possibilities:

Possibilities for conjunctions and disjunctions of propositions are calculated using operators which are not necessary the same as those of probability. A popular choice is to use the MIN operator for conjunction and the MAX operator for disjunction. For example, the possibilities for the conjunction and disjunction of the propositions p1 and p1 may be calculated as follows:

POS(p1 AND p1) = MIN (POS(p1), POS(p2)),

POS(p1 OR p1) = MAX (POS(p1), POS(p2)).

The possibility of (NOT p) is given by:

POS(NOT p1) = 1.0 - POS(p1).

However, these rules for combination are not generally valid. The MIN/MAX combination rules are only strictly valid when p1 and p2 are mutually dependent:

p --> q or q --> p

but in practice these conditions are often ignored and the calculus used without proper justification.

For example, if p1 and p2 are mutually exclusive propositions then POS(p1 AND p2) should be zero, but the above rules will incorrectly assign the minimum of POS(p1) and POS(p2). Care must be taken to avoid this type of error.

3.3 Support Logic.

Support Logic is a generalisation of probability which also includes a method of representation and reasoning with fuzzy sets. It overcomes the problem of too restrictive a calculus by representing uncertainty by a pair of evidential supports which can be interpreted as an interval in which the unknown probability lies. The theory was developed by Jim Baldwin [Baldwin, 1986, 1987] at Bristol University where it continues to be extended.

Support Logic has the following significant features:

1) well-defined theory for modelling heuristics,

2) open world semantics,

3) semantic unification, and

4) generalisation of probability via support pairs.

Support Logic has also been incorporated into a declarative programming language called FRIL [Baldwin, 1987].

FRIL provides a PROLOG language [Clark & McCabe, 1984; Clocksin & Mellish, 81] which enables representation and inference under probabilistic uncertainty and fuzziness.

The basic ideas of Support Logic will be presented in the following three sub-sections focusing on the representation of uncertainty, combination of uncertainty and inference under uncertainty.

Following these there will be a short section outlining semantic unification via fuzzy sets, and then a brief section describing the FRIL language.

3.3.1 Support Logic representation of uncertainty.

Support Logic represents uncertainty by a pair of supports which are referred to as a 'support pair'. For example, the uncertain evidence relating to the truth of a proposition p is represented:

p : [l, u].

where l and u are known respectively as the necessary and possible supports.

The 'necessary support' l represents evidence in support of the proposition, and 1.0 minus the 'possible support' (1.0-u) represents evidence supporting the negation of the proposition. The difference between the necessary and possible supports (u-l) represents the evidential uncertainty. This is evidence which supports neither the proposition nor the negation of the proposition.

The support for the negation of p is given by:

~p : [1.0-u, 1.0-l].

It is helpful to consider two interpretations of the 'support pair'. The first and most simple (already mentioned above) is that of a probability interval. The necessary and possible supports are interpreted as the lower and upper bounds of the range in which the probability is thought to lie, based on the contributing evidence.

The second interpretation of the support pair is illustrated by the following example. Consider a random sample of the population answering the question:

'Will nuclear energy plans be an important issue in the next General election ?'

Imagine the response: 55 % vote yes
35 % vote no
10 % abstain (not sures).

The necessary support for nuclear energy being an important issue is 0.55, and the possible support is 0.65. The evidential uncertainty is 0.1. This could be represented as:

(important_issue nuclear_energy) : [0.55, 0.65].

3.3.2 Support Logic combination of uncertainty

Support logic [Baldwin, 1986, 1987] provides three methods of calculating conjunction and disjunction. The choice of which method to use is made on the basis of the interdependence of the
conjuncts/disjuncts.

Consider a pair of propositions p1 and p2 with supports:

p1 : [l1, u1]

p2 : [l2, u2].

The aim is to determine supports for the compound propositions:

c = (p1 AND p2),

d = (p1 OR p2).

The following table relates the necessary supports m_{ij} for the various statements involving p1 and p2:

			l2	1.0-u2	u2-l2
			p2	~p2	p2 # ~p2
l1	p1	|	p1 & p2	p1 & ~p2	p1
		|	m_{11}	m_{12}	m_{13}
1-u1	~p1	|	~p1 & p2	~p1 & ~p2	~p1
		|	m_{21}	m_{22}	m_{23}
u1-l1	p1 # ~p1	|	p2	~p2	uncertain
		|	m_{31}	m_{32}	m_{33}

N.B. The symbols (pi # ~pi) represent the evidential uncertainty about pi (i=1,2).

The supports m_{ij} can be interpreted as 'fluid' probabilities which can be adjusted within the constraints imposed by row and column totals within the table.

The constraints on the m_{ij} are as follows:

$$m_{11} + m_{12} + m_{13} = l1$$

$$m_{21} + m_{22} + m_{23} = 1.0-u1$$

$$m_{31} + m_{32} + m_{33} = u1-l1$$

$$m_{11} + m_{21} + m_{31} = l2$$

$$m_{12} + m_{22} + m_{32} = 1.0-u2$$

$$m_{13} + m_{23} + m_{33} = u2-l2$$

$$m_{11} + m_{12} + m_{13} + m_{21} + m_{22} + m_{23} + m_{32} + m_{33} = 1.0.$$

These constraints do not give unique values for the m_{ij}, but interval constraints on all the m_{ij} can be deduced, for example:

$$MAX(0.0, l1+l2-1.0) \quad =< \quad m_{11} \quad =< \quad MIN(l1, l2)$$

which corresponds to the relation:

$$MAX(0.0, PR(p1)+PR(p2)-1.0)$$
$$=< PR(p1 \text{ AND } p2)$$
$$=< MIN(PR(p1) , PR(p2)).$$

Note that the choice of PR(p1 AND p2) = PR(p1) * PR(p2) which corresponds to the assumption of independence in probability theory, satisfies the constraint relation expected.

Multiplication model

In the absence of any information, it assumed that p1 and p2 are independent, so that:

$m_{11} = l1 * l2$

$m_{12} = l1 * (1.0-u2)$

$m_{21} = (1.0-u1) * l2$

$m_{22} = (1.0-u1) * (1.0-u2)$

whence:

(p1 AND p2) : [Sl1*Sl2, Su1*Su2]

(P1 OR p2) : [(Sl1+Sl2-Sl1*Sl2), (Su1+Su2-Su1*Su2)].

Conservative model

This model takes a pessimistic view on the available information and therefore takes the widest possible interval for inferences consistent with the constraints on the $m_{ij,}$ whence:

(p1 AND p2) : [MAX(0.0, l1+l2-1.0), MIN(u1, u2)]

(p1 OR p2) : [MAX(l1, l2), MIN(1.0, u1+u2)]

Strict implication

This model assumes that either p1 strictly implies p2 or conversely. This corresponds to strict dependence between the propositions p1 and p2, in contrast to the multiplication model, the expressions obtained are:

(p1 AND p2) : [MIN(l1,l2), MIN(u1,u2)]

(p1 OR p2) : [MAX(l1,l2), MAX(u1,u2)]

Combination of evidence from multiple proof paths

A support logic program may use many different paths by which to establish a particular conclusion. In a logic programming language such as PROLOG the interpretation of the truth of a goal is relatively simple because a goal which is true by one path must also be true by any other. However, in support logic there may be many paths to a goal, each with an associated support pair. All such support paths have to be combined to give an overall support pair for the goal.

Two different methods for combination of supports are defined in support logic, corresponding to:

1) mutually dependent proof paths, in which the support pairs from different proof paths are intersected together, and

2) mutually independent proof paths, in which a 'conflict resolution' rule is used to resolve the conflict between potentially conflicting viewpoints.

In practice, the 'mutual independence' rule is most commonly applicable and is therefore the default combination rule used by FRIL. As the following example shows, this type of combination is able to extract precision from uncertainty and redundancy. Consider the following support logic inferences for a conclusion c:

c : [0.4, 0.7]

c : [0.6, 0.9].

The 'mutual independence' rule gives the combined support as:

c : [0.4, 0.7].

This is more precise than the either of the contributory supports and can be justified by the requirement that PR(c) must lie within both the intervals [0.4,0.7] and [0.6,0.9] simultaneously. A null intersection would indicate an inconsistency in the knowledge.

The second method of combining support pairs is given by the 'conflict resolution' rule. This is based on Dempster's rule, and is used to combine evidence from independent viewpoints. In this case, support pairs need not overlap and may in fact be at considerable variance.

The 'conflict resolution' rule combines the supports [n1,p1] and [n2,p2] to give an overall support pair [n,p] as follows:

n = (n1 + n2 - n1 * n2 - c) / (1.0 - c)

p = (p1 * p2) / (1.0 - c)

where c represents the conflict between the support pairs representing the two points of view and is given by:

c = n1 + n2 - n1 * p2 - n2 * p1.

3.3.4 Support Logic inference

Consider the inference is characterised by:

p --> q : [Cl,Cu]
p : [pl,pu]
---
q : [ql,qu],

where [Cl,Cu] is the conditional support pair for q given p, ie:

Cl =< PR(q|p)=< Cu.

The problem is to deduce values for the support for q on the basis of the support for p and the support for q given p.

The theorem of total probabilities is as follows:

PR(q) = PR(q|p) * PR(p) + PR(q|~p) * PR(~p).

This can be used to derive the support for q:

ql = Cl * pl + 0.0 * (1.0-pu)

= Cl * pl

qu = Cu * Pu + 1.0 * (1.0-pl)

= Cu * pu + 1.0 - pl.

An extended form of the inference rule includes a conditional support pair for q given (NOT p). The inference problem then becomes:

p --> q : [Cl1,Cu1] [Cl2,Cu2]
p : [pl,pu]
---
q : [ql,qu].

where [Cl1,Cu1] is the conditional support pair for q given p and [Cl2,Cu2] is the conditional support pair for q given (NOT p).

With this additional evidence the total probability rule gives the inferred support pair for q to be:

ql = Cl1 * pl + Cl2 * (1.0-pu)

qu = Cu1 * pu + Cu2 * (1.0-pl).

This clearly gives a narrower interval for the probability of q.

3.3.4 Fuzzy sets and semantic unification

PROLOG provides only a syntactic form of unification. That is, two terms can match only if substitutions can be made to make the two terms equivalent symbol by symbol. However it is often useful for knowledge representation in expert systems to characterise terms by semantic labels.

For example if a person is described as being of 'above average height', then there must be some support for that person being 'tall'. In order to derive support in this way, semantic rather that syntactic unification is needed.

The semantic labels 'tall' and 'above_average_height' are characterised by fuzzy set definitions expressed as possibility distributions (ie. set membership functions) on a continuous 'height' domain.

It is not intended to go into details of the method in this work. However, the basic idea behind the method is as follows. A possibility value for the distribution 'above_average_height', and the distribution itself gives rise to a possibility distribution for the height of the person. If this is used in conjunction with the 'tall' possibility distribution, it gives rise to a possibility value for the person being 'tall'.

3.3.5 The language FRIL

FRIL [Baldwin, 1987] is a logic programming language based on Support Logic [Baldwin, 1986, 1987]. It first became available within the Information Technology Research Centre of University of Bristol during 1986. Unfortunately, this was not soon enough to be of use in implementing the turbo-generator expert system.

All of the features of Support Logic so far described are supported by the FRIL language, which is available for a range of machines and operating systems.

FRIL is most easily described as a generalisation of PROLOG which allows probabilistic and fuzzy reasoning incorporating semantic unification.

As in PROLOG, inference is performed by backward chaining from the goal to the facts.

FRIL chooses its search mechanism depending on the type of reasoning being performed. To solve a PROLOG type problem (where no clauses have support pairs), FRIL uses a search strategy identical to that of PROLOG, ie. depth-first with backtracking. To solve a problem involving uncertainty (where some of the clauses have support pairs), FRIL uses a mixture of breadth-first search and depth-first search with backtracking.

3.4 A 'new' model of uncertainty

The uncertainties of turbo-generator state monitoring are a powerful motivation for deriving a model for reasoning under uncertainty.

The knowledge representation must be clear and concise to facilitate knowledge acquisition and maintenance. It must also be sufficiently expressive to accommodate wide ranging uncertainty and still be capable of deriving realistic overall support for conclusions.

These requirements would tend to indicate the need for an economical but precise knowledge representation that can accommodate the different types of uncertainty within a single, clearly defined framework.

Chapter 4 will describe in detail a method for knowledge representation and inference under uncertainty. This method is similar in many respects to Support Logic (described above) but is slightly more rigid although rather more concise and economical.

The 'new' model of uncertainty evolved during the process of knowledge elicitation from experts at the CEGB, and out of a need for a representation of uncertainty that was more flexible than that of possibility theory which at first appeared to be suitable.

The 'new' model is most easily explained by relating it to Support Logic theory which was proposed by Jim Baldwin [Baldwin, 1986,1987]. The 'new' model of uncertainty can be described as follows.

Evidential uncertainty is represented by a pair of numbers, as in support logic. For example, the weight of evidence relating to piece of knowledge p is represented as:

k : [l,u]

where [l,u] is equivalent to a support pair.

k can be:

1) a single proposition,

2) a logical combination of propositions, formed by either conjunctions or disjunctions of propositions,

3) a rule of inference of the form: p --> q, which states that if p is true then q is also true.

Combination of uncertainty within conjunctions and disjunctions is equivalent to that used by default in support logic (the multiplication model described in section 3.3.2).

In the following, and in chapter 4 which describes the new knowledge representation in full, the rule of inference:

p --> q : [l, u]

will be denoted:

q :- p : [l, u]

which reads:

'q is true' if 'p is true' : [l, u].

The overall method of fault diagnosis is based a generate and test paradigm. Hypotheses are generated by referring to key symptoms and are then tested using both refutative and corroborative approaches.

The two approaches to fault hypothesis testing are considered in combination to give an overall assessment of hypotheses, but opposing and supporting evidence is derived by independent inference from the symptoms.

3.4.1 Refutative testing of fault hypotheses

Refutative testing of fault hypotheses is based on reasoning from rules which specify necessary conditions, ie. symptoms which are to some extent guarantied to be present when the fault exists. These rules are of the form:

F :- S1 OR S2 OR ... OR Sn : [0.0,u].

The effect this rule is to reduce the inferred possible support for F by an amount which depends on how low the combined possible support for the body of the rule is, and on how low the conditional possible support u is.

Rules of this form are a special case of support logic rules and are therefore amenable to direct implementation in the standard FRIL language.

3.4.2 Corroborative testing of fault hypotheses

Corroborative testing of fault hypotheses is based on reasoning from rules which specify possible or characteristic symptoms associated with faults.

The basic idea is to assess the weight of evidence for the existence of a fault by considering the supporting evidence for characteristic symptoms of the fault.

Corroborative evidence for a fault hypothesis F is inferred using support logic rules of the form:

F :- S1 AND S2 AND ... AND Sn : [1,1.0].

The effect of this rule is to assign a necessary support for F, based on the weight of necessary support associated with the conjunction of symptoms S1,S2,...,Sn.

Rules of this form are also a special case of support logic rules, and are therefore amenable to direct implemented in the FRIL language.

An obvious weakness of this type of rule is that if one of more of the symptoms Si (i=1..n) has a low necessary support then the rule will return a support pair which is identical or close to [0.0,1.0]. In doing this, the effect of the rule would be to waste the evidence available in well supported symptoms.

The new model of uncertainty overcomes this problem by using an indirect method of representation for corroborating rules. The necessary knowledge for constructing the rule is represented directly, and in order to perform inference, a rule is constructed to maximise the use of available support.

Construction of the rule involves:

1) choosing a set of conjuncts for the body of the rule, and

2) calculating the rule's conditional support pair on the basis of the potential weight of evidence for the fault F which could be derived from the chosen conjuncts (symptoms).

CHAPTER 4

KNOWLEDGE REPRESENTATION AND INFERENCE

4.1 Introduction.

This chapter details a knowledge representation and method of inference for use in turbo-generator fault diagnosis. Faults, symptoms, and expert knowledge are represented within a single framework that facilitates evidential reasoning.

The following contains details of the method, and describes how it enables diagnoses based on large numbers of uncertain symptoms.

The chapter is divided into 3 main sections:

1) fault and symptom representation (section 4.2),
2) diagnostic knowledge representation, and inference method (section 4.3),
3) remedial action(s) selection (section 4.4).

It is assumed that the reader has a reasonable understanding of 'support logic'. The previous chapter presents basics of the language but reference material can be found in [Baldwin, 1986, 1987].

Chapter 2 explains the problem of turbo-generator state monitoring. If background information is required then the user should refer back to this chapter.

An important aspect of turbo-generator state monitoring is to be able to diagnose faults on the basis of symptoms. The representation of faults and symptoms is described in the following section.

4.2 The representation of faults and symptoms.

Faults and symptoms are represented as predicates of the form:

object (LABEL, FULL_NAME, SHORT_NAME, TYPE)

where:

LABEL indicates whether the object is high level **h** or low level **l**. High level means that evidence about the object derives from lower level reasoning, and low level means that evidence derives from direct observation.

FULL_NAME is the full name of the object, for example **non-axisymmetric rotor cracking**.

SHORT_NAME is a shortened version of FULL_NAME, for example **na_crack**.

TYPE indicates the nature of the object; whether it is a fault or a certain type of symptom.

Fairly complete sets of faults and symptoms, represented in this form, are given in appendix A.

Example of fault representation:

Loss of mass is a fault, or rather a class of fault. It describes a situation in which a rotating part of the machine, part of a rotor for example, falls off. This may indicate the that part of the machine is disintegrating and may also cause problems by virtue of a resulting out of balance.

The fault **loss of mass** is represented:

object (**h, loss of mass, lom, fault**)

where:

a) **h** indicates that evidence relating to the existence of **loss of mass** derives from lower level reasoning, based on more readily observable objects;

b) **loss of mass** is the name of the object;

c) **lom** is an identifier for **loss of mass**; and

d) **fault** indicates that the object **loss of mass** is a fault.

Example of symptom representation:

The symptom **step change** describes a change in a machine vibrations which occur apparently, instantaneously.

Step change is described as a 'high level' symptom because evidence which relates to it, derives from a lower level, ie. from the time taken. The time taken for the change to occur is input to the system. If this is close to zero then the system uses inbuilt knowledge to derive evidence supporting a step change.

The symptom **step change** is represented:

object(h,**step change**,**step**,*temporal*)

where :

a) h has the function indicated in a) immediately above;

b) **step change** is the name of the object;

c) step is an identifier for **step change**; and

d) *temporal* indicates that the object is of type *temporal* (i.e. relating to the correlation of changes with time).

The low level symptom **time taken** is represented:

object(l,**time taken'**,*time*,*temporal*)

where:

a) 1 indicates that evidence relating to the existence of **time taken** is not to be calculated from lower level symptoms, but is to be assessed directly by an observer;

b) **time taken** is the name of the object;

c) time is an identifier for **step change**; and

d) *temporal* indicates that the object is of type *temporal* (i.e. relating to the correlation of change with time).

4.3 Fault diagnosis from symptoms.

4.3.1 Representation of diagnostic knowledge.

Within the proposed scheme, knowledge is represented using first order predicate logic. To accommodate uncertainties in the evidence relating to symptoms and expert knowledge, uncertainty measures are attached.

The uncertainty measures used by this scheme can be regarded as 'support pairs' as defined in support logic which is an evidential model of uncertainty described in chapter 3.

Each piece of knowledge, symptom or fault P, has an associated support pair:

P : [l,u]

where [l,u] represents the range in which the probability of P being true is believed to lie.

l is known as the necessary support for P, and represents the weight supporting evidence for P.

u is known as the possible support for P, and (1.0-u) represents the weight supporting evidence for (NOT P).

See chapter 3 for a description of support logic.

The following describes the uncertainty measures and their combination during inference. The scheme's principal aim is to accommodate different types of uncertainty within a single concise framework.

A large part of the chapter addresses the problems of reasoning with incomplete data. Where diagnostic data derives from multiple sources a flexible knowledge representation is essential. If knowledge is improperly structured, missing data will render the diagnosis ineffective by propagating uncertainty. This is clearly so for a rigid support logic rule of the form:

a :- **x,y,z** : [csl, csu]

which means that the probability that:

'**a** is true if **x** and **y** and **z** are all true'

is between csl and csu.

According to support logic, the derived uncertainty in '**a**' will be at least as large as that in either x,y or z. Therefore if any one of x,y,z is unknown the rule is effectively useless. In large rigid rules, containing perhaps 10 conjuncts, it would be almost inevitable for one or more to be uncertain. Therefore such a rule would almost never be of use.

A major aim of the model of uncertainty which is about to be described, is to overcome this problem.

4.3.2 The approach to fault diagnosis.

The approach to fault diagnosis is based on a 'generate and test' paradigm, and uses a method similar to 'support logic' to model uncertainty.

A generate and test approach to diagnosis (as explained in chapter 2) generates hypotheses (by forward chaining from symptoms), then tests them (using backward

chaining from the hypotheses to the symptoms). Expert knowledge is thus divided into groups corresponding to 'hypothesis generation' and 'hypothesis testing'.

Hypothesis generation.

Hypothesis generation uses two complimentary methods. One uses apriori machine knowledge, the other uses expert knowledge in conjunction with machine observations.

The method creates hypothesis sets represented respectively by **H1** and **H2**. The set union, represented by **H** (= {**H1** ; **H2**}), is used as the working hypothesis set.

The first hypothesis generation method relies on apriori knowledge about the machine. For example, known machine weaknesses can provide clues about likely (or possible) faults. Machine maintenance work may also provide clues regarding machine behaviour changes, (since some maintenance work can trigger off certain faults).

Even when unusual machine behaviour is reported without details, an expert will often be able to suggest possible faults. Apriori knowledge is clearly being used.

Given no behavioural details, the most logical faults to hypothesise, are the most common machine faults.

The second method of hypothesis generation uses expert knowledge about turbo-generators. This comprises knowledge of varying scopes, valid for individual machines, designs of machine and machines in general. Machine observations are considered in the light of expert knowledge about how turbo-generators are expected to behave (in the event of certain faults or operating conditions). This method differs from the first method which generates hypotheses in isolation from details of machine behaviour.

Key symptoms can usually be identified, thus implicating possible faults. A crude causal model can provide links between the symptoms and their possible faults.

To restrict the working hypothesis set size, a single machine fault is assumed. This will usually be proven justified when a fault is successfully diagnosed. If there are

multiple real faults, the hypothesis set may not include some of them. For example, suppose the key symptoms SK1 and SK2 are evident. If their corresponding sets of possible faults are F1 and F2, assuming a single fault, the intersection of F1 and F2 should contain the true fault. Assuming multiple faults, the union of F1 and F2 can be used.

The above approach takes no account of the weight of evidence which is associated with key symptoms but could easily be modified to so.

Hypothesis testing.

Hypothesis testing combines two complimentary approaches; negative and positive testing. The two are applied independently thus generating a support pair for each fault. Corresponding support pairs are combined to give an overall measure of support for each fault.

The aim of negative testing is hypothesis refutation. This is accomplished using evidence supporting the negation of expected symptoms. For example:

a implies b
there is evidence for (NOT b)
---
deduce: there is evidence for (NOT a).

Positive testing aims to confirm hypotheses by using supporting evidence of machine behaviour. For example, if the symptoms $a_1,a_2,...,a_n$ positively correlate with a certain fault, then the presence of any subset of them will provide a certain level of positive support. A large part of this work is concerned with assessing such levels of support, ie. in assessing the overall support provided by large numbers of partially supported symptoms.

4.3.3 Representation of knowledge for negative testing.

Negative testing knowledge, is represented by rules of the form:

a ===> b : [CNS,1.0]

indicating that if a is true, b is true also. The rule can be alternatively written as:

NOT a :- Not b : [CNS,1.0]

evidence for 'NOT a' derives from evidence for 'NOT b'.

The upper conditional support being 1.0, restricts use of the rule to refutation of 'a' (see chapter 3).

The lower conditional support CNS has a value between 0.0 and 1.0. The higher CNS, the higher the potential refutation of 'a'. For example, if CNS = 0.0 the rule is means that 'a' does not necessarily imply 'b'. Thus, evidence against 'b' cannot be used as evidence against 'a'. Conversely, if CNS = 1.0 the evidence of 'NOT b' is equal evidence of 'NOT a'.

CNS acts (intuitively) as a weighting factor, moderating the effect of evidence for 'NOT b', on 'NOT a'. Rewritten in standard support logic notation, the above rule becomes:

a :- b : [0.0, CPS]

where CPS = 1.0 - CNS.

Let NS(x) and PS(x) denote the necessary and possible supports for x. The above rule can be alternatively written:

NS(a) = 1.0 - (1.0 - PS(b)) * (1.0 - CPS)

PS(a) = 1.0

or using the first form of the equation:

NS(a) = 1.0 - (1.0 - PS(b)) * CNS.

If CNS = 0.0, NS(a) = 1.0 irrespective of PS(b).

If CNS = 1.0, NS(a) = PS(a).

For intermediate values of CNS, NS(a) will have a value dependant on PS(b) and CNS.

Refutation or 'negative testing' knowledge is intuitively represented by the rule:

fault X ===> machine behaviour associated with X.

This means that the machine is expected to behave in certain ways if fault X exists. If it does not, there is evidence that the fault does not exist. The problem is how to represent this knowledge in a concise but meaningful way.

One possibility would be to produce a separate rule:

fault F_i ===> symptom S_j : [NCS_{ij}, 1.0]

for each symptom S and fault F. Each symptom would be used independently to refute each fault. Given n faults and m symptoms, n * m rules would be generated. Of course, many of these rules would be trivial, of the form:

fault F ===> symptom S : [0.0, 1.0]

since many symptoms will be unrelated to a given fault.

In practice such rules would perform reasonably most of the time. They do not however, account for the interdependence of symptoms, and may therefore not produce overall realistic supports. For example, the rule (temporarily ignoring supports):

F ===> S1 or S2 -----(R1)

cannot be adequately represented by the rules:

F ===> S1

F ===> S2.

The first rule assumes S1 and S2 are dependent whilst the second rule assumes they are independent.

If some additional flexibility is added to allow rules of the similar to (R1), then such problems are more easily avoided.

A further extension to the scheme, allowing multi-level rules will greatly reduce the number of rules required. For example:

F ===> A

F ===> B

A ===> S1

A ===> S2

A ===> S3

B ===> S4

B ===> S5

where A and B are not directly observable symptoms.

Consider a further example **of out of balance** behaviour which is expected to result from the fault **loss of mass**. However, **out of balance** is only observable indirectly, via observable symptoms. Thus two levels of rule can be useful. The intermediate stage could be eliminated, but it is useful to keep for three reasons. Firstly, because it reduces the number of required rules (rules forming a tree structure). Secondly, multi-level knowledge is easier to collect from experts, because it is more natural. Thirdly, explanation (of diagnosis) is easier where represented knowledge closely resembles (intuitive) expert knowledge.

An extensive example of the above refutation scheme is presented later in the chapter.

The above refutation knowledge is based on expectation. For each fault certain types of behaviour are expected. Evidence that there is no such behaviour, is evidence that the fault does not exist.

The basis for these expectations comprises two main elements: scientific and experiential knowledge.

Experts involved in turbo-generator monitoring generally have a reasonable knowledge of Newtonian mechanics. This forms the basis of their understanding of turbo-generator behaviour.

Their knowledge, associates a causal model with the turbo-generator. This predicts certain machine behaviour for hypothesised faults. Differences between predicted and actual behaviour enable hypothesis refutation.

The above knowledge is supplemented by experiential knowledge, otherwise referred to as heuristic or rule of thumb. This is just as useful for refutation, although sometimes more difficult to justify.

When combined, scientific and experiential knowledge provide a powerful basis for negative hypothesis testing.

The following example should ensure a fuller understanding of the proposed representation.

<u>The method of inference using knowledge for negative testing hypotheses.</u>

The method of inference which is applied to knowledge represented in the above form, is that dictated by support logic. An illustration of the method of inference is presented in the example below.

Let the following fault and symptom mnemonics be defined:

mnemonic = name of fault.

trotcrack = non-axisymmetric crack,

stas = stiffness asymmetry,

tprtrend = trend in the twice per rev,

ssmcs = changes in the signature at critical speed and sub-multiples thereof,

sfsr = changes in the signature over the whole speed range.

knowledge base:

```
trotcrack ===> stas       : [1.0, 1.0].    ----(W)
trotcrack ===> tprtrend   : [0.6, 1.0].    ----(X)
stas      ===> ssmcs      : [1.0, 1.0].    ----(Y)
stas      ===> sfsr       : [0.7, 1.0].    ----(Z)

ssmcs     : [0.8, 0.8].
sfsr      : [0.5, 0.5].
tprtrend  : [1.0, 1.0].
```

end of knowledge base.

Rules of the form 'a ===> b :[l,u]' state that the probability that 'b is true if a is true' is within the range l to u. In support logic notation, these rules are usually written:

a :- (NOT b) : [(1.0-u),(1.0-l)].

Similarly, a fact of the form 'a :[l,u]' states that the probability that 'a is true' is within the range l to u.

rewriting (X) in its usual form:

trotcrack :- NOT **tprtrend** : [0, 0.4],

so POS(trotcrack) = 1.0 - (1.0-POS(trotcrack|NOT tprtrend))
*NEC(NOT TPRTREND),
= 1.0-(1.0-0.4)*(1.0-POS(tprtrend)),
= 1.0-(1.0-0.4)*(1.0-1.0),
= 1.0.

NEC(trotcrack) = 0.0

trotcrack : [0.0, 1.0]. ----(1)

rewriting (Y):

stas :- NOT ssmcs : [0.0, 0.0],

so POS(stas) = 1.0-(1.0-0.0)*(1.0-0.8),
= 1.0-0.2,
= 0.8.

NEC(stas) = 0.0.

stas : [0.0, 0.8]. ----(2)

rewriting (Z):

stas :- NOT sfsr : [0.0, 0.3],

POS(stas) = 1.0 - (1.0-0.3)*(1.0-0.5),
= 1.0 - 0.7*0.5,
= 0.65.

NEC(stas) = 0.0.

stas : [0.0, 0.65]. ----(3)

using Dempster's combination rule on (2) and (3):

stas : [0.0, 0.52]. ----(4)

then from (W):

trotcrack :- NOT stas : [0.0, 0.0],

POS(trotcrack) = 1.0-(1.0-0.0)*(1.0-0.52),
= 0.52.

NEC(trotcrack) = 0.0

trotcrack : [0.0, 0.52]. ----(5)

then using Dempster's combination rule on (1) and (5):

trotcrack : [0.0, 0.52].

4.3.4 The representation of knowledge for positive testing.

Positive hypothesis testing is performed using knowledge of the form:

fault :- **behaviour.** ----(A)

This is modelled using rules of the form:

fault :- **conj** : [CNS, CPS]. ----(B)

where:

1) **fault** is a hypothetical fault,
2) **conj** is a conjunction of symptoms,

3) [CNS, CPS] is the conditional support for **fault**, given **conj**.

The method of positive hypothesis testing uses all available supporting evidence. Supporting evidence must be considered as a whole, to give an overall realistic support. The idea is expressed simply by rule (A) above.

The difficulty in modelling positive testing is choice of suitable rules.

A simple approach is to construct single conjunct rules of the form:

fault :- symptom

These rules would have a typically low conditional necessary support. The supports resulting from each rule would be similarly low, but combine with each other to produce a larger overall support. However, this approach ignores interdependence of symptoms, and the resulting overall support generally underestimates the true support.

The problem with such rules can be partially overcome by manipulating the conditional supports on the rules. This should avoided if possible, because it complicates knowledge acquisition and explanation analyses.

Another modelling approach would be to have just one rule of the form:

fault :- CONJ : [CNS, CPS].

where **conj** is the conjunction of all **fault** related symptoms.

Given full support for each symptom in **conj** it should be possible to derive full support for **fault**. However, in practice **conj** will contain several symptoms (10 perhaps), and the rule will therefore almost always return complete uncertainty:

fault : [0.0 1.0]

The reason for this, is the very high probability of at least one of the symptoms being uncertain. For example, the measurements needed to assess the support for some of the symptoms may not have been made.

The two above approaches (several small rules or one large one) are opposite extremes in the range of possible approaches. The proposed scheme is a compromise between them.

The problem encountered with the latter method (of having a single large rule) can be overcome by including all possible alternative rules. For example, if **conj** = (U,V), then the main rule would be:

fault :- U,V : [CNS, CPS].

The full set of rules would then be:

fault :- U,V : [CNS, CPS].

fault :- U : [CNS1, CPS1].

fault :- V : [CNS2, CPS2].

If, for example, evidence for the symptom U were uncertain then the second rule would apply. In standard support logic, this method is implemented by taking the intersection of the supports which derive from each of the rules. The effect is similar. For example, given supports for U and V, each above rule yields a support pair:

fault : [CNS, CPS]
fault : [CNS1, CPS1]
fault : [CNS2, CPS2]

Taking the intersection of these support pairs gives the overall support as:

[N, P]

where:

N = maximum of (CNS CNS1 CNS2), and

P = minimum of (CPS CPS1 CPS2).

As a further illustration of the above, consider the rule:

heavy_person :- tall_person, fat_person : [CNS, CPS].

This expresses the idea that a tall, fat person will be heavy. Given no information about height and therefore complete uncertainty about tall_person, the rule will yield complete uncertainty for heavy_person. Clearly this is not ideal, since fat_person alone should provide some support for heavy_person. Knowledge to this effect must be explicitly stated:

heavy_person :- fat_person : [CNS1, CPS1]

and similarly:

heavy_person :- tall_person : [CNS2, CPS2].

Whatever the evidence for tall_person and fat_person, support logic will assign a reasonable support to heavy_person.

This approach works well in cases like the ones above where there are only small conjunctions of symptoms. However, when the number becomes larger it becomes impractical to derive and store the large number of necessary of rules.

The proposed knowledge representation scheme uses an approach similar in some respects to enumeration approach but stores knowledge in a more concise way.

Evidence for each fault, derives from a number of different sources, ie. from:

temporal,
locational,
frequency,
speed,
non-vibrational,
external,

aspects of machine behaviour. These can be used in combination to describe the manifestions of each fault, and therefore the faults themselves. The above will henceforth be referred to as descriptions.

The above descriptions have been chosen to encompass all aspects of machine behaviour. Using them it should be possible to uniquely define a fault by virtue of the behaviour it produces.

Thus, each fault is described by its *temporal*, *locational*, *frequency etc* behaviour. The *temporal* behaviour of the fault "non-axisymetric rotor crack" (**na_crack**) is denoted :

na_crack (*temporal*).

Positive diagnostic knowledge for **na_crack** could be expressed:

na_crack :- **na_crack** (*temporal*),
na_crack (*locational*),
na_crack (*frequency*),
na_crack (*speed*),
na_crack (*non-vibrational*),
na_crack (external) : [1.0 1.0].

ie. full positive evidence (necessary support) for **na_crack** derives from the six aspects of its characteristic behaviour.

An important addition to the above, is a measure of the relative importance of the descriptions. For example, evidence of *temporal* behaviour is especially important to

crack diagnosis, and is therefore weighted more strongly than evidence of other behaviour. Some types of behaviour are of no importance whatever to crack diagnosis, and related evidence therefore has a zero weighting.

The proposed weighting scheme uses one number per description. This number is lies in the range 0.0,1.0 and is referred to as a *threshold*. The more important the description, the lower its threshold value.

A description essential to diagnosis, has a threshold value of 0.0, whereas an irrelevant description has a threshold value of 1.0.

The value of a description's *threshold* has a simple interpretation. It is the maximum necessary support for the fault (in an open world) that can be derived from all evidence unrelated to the description. For example, if:

na_crack(*locational*)

represents a type of behaviour described as *locational*, whose associated threshold is 0.4 then, if this behaviour is not considered, the maximum necessary support that can be assigned to **na_crack**, is 0.4 (irrespective of other evidence).

The above weighting scheme will be more fully discussed later in the chapter.

The evidence for each component of behaviour d_i, is calculated using a set of support logic rules which will be referred to as measures (ie. of the description).

Each d_i has an associated set of rules that are used to measure the weight of supporting evidence. For example:

na_crack(*locational*) :- **A,B** : [1.0 1.0]
na_crack(*locational*) :- **A** : [0.3 1.0]
na_crack(*locational*) :- **B** : [0.2 1.0]
na_crack(*locational*) :- **true** : [0.0 1.0]

The support they provided for **na_crack** by the *locational* measures depends on the threshold value T of the description *locational*:

T = threshold(*locational*).

The aim is to choose the measure that provides most supporting evidence when combined with evidence from complementary descriptions.

For example, consider the case where both **A** and **B** have a [0.0 1.0] support. It would be wrong to use a [0.0,1.0] support directly in conjunction with support from other descriptions since the resulting overall support would be [0.0,1.0]. This is obviously inappropriate since the threshold value of *locational* is 0.4.

The evidence provided by the *locational* description ought not to detract from the potential necessary support of complementary descriptions. The overall support should be at least 0.4.

The above is further illustrated in the following example.

Illustration of representation scheme.

Consider the knowledge required for positive diagnosis of a non-axisymmetric rotor crack:

FAULT = **non-axisymmetric cracking**

DEGREE OF DEFINITION = 1.0

DESCRIPTION	THRESHOLD
speed	0.1
locational	0.2
external	0.8
temporal	0.5

This shows that **non-axisymmetric cracking** is described by: *speed*, *locational*, *external*, and *temporal*. The extent to which these combined descriptions define **non-axisymmetric crack** is given by:

DEGREE OF DEFINITION (= 1.0).

This is the maximum necessary support for **non-axisymmetric cracking** that can derive from the combined evidence of *speed*, *locational*, *external*, and *temporal* aspects of machine behaviour.

The degree of definition equals 1.0, because the descriptions cover all relevant evidence. However, if *temporal* evidence (for example) were missing, the remaining degree of definition would be reduced to 0.5 (the threshold value for *temporal*).

The threshold values, represent the importance of each description. The *speed* and *external* aspects of behaviour have threshold values of 0.1 and 0.8 respectively. Thus *speed* behaviour is more important than *external* behaviour, to positive diagnosis. As explained earlier:

threshold(*external*) = 0.1

indicates the maximum necessary support for the fault that can derive from evidence other than *external* behaviour.

Additional knowledge is required, providing details of how to assign support for each aspect of behaviour. For example, support for:

na_crack (*speed*)

derives from the symptoms:

sfsr =

changes in vibrations over the whole speed range (measured during a run-up/down),

ssmcs =

changes in vibrations at critical speeds and sub-multiples,

ccs =

changes in critical speeds.

The measure of support assigned to **na_crack** (*speed*) is determined by:

na_crack (*speed*):-	*1*	**sfsr, ssmcs, ccs**	: [1.0, 1.0].
	2	**sfsr, ssmcs**	: [0.9, 1.0].
	3	**sfsr, ccs**	: [0.3, 1.0].
	4	**srf**	: [0.2, 1.0].
	5	**ssmcs, ccs**	: [0.7, 1.0].
	6	**ssmcs**	: [0.5, 1.0].
	7	**ccs**	: [0.1, 1.0].
	8		: [0.0, 0.0].

These are referred to as *measures* of the behaviour **na_crack**(*speed*). Each is a rule comprising head, body and conditional support pair. The heads are the same for each measure, but the bodies and support pairs different. The bodies form a complete set of alternatives.

The conditional necessary supports reflect the ability of the rules' body to measure the *speed* aspect of the fault's description. Another way of looking at these rules is as follows.

The effect of **na_crack** on the *speed* aspect of machine behaviour can be characterised using the symptoms **sfsr**, **ssmcs**, and **ccs**. A [1.0 1.0] support for each of these, would provide a [1.0 1.0] support for **na_crack**(*speed*).

However, in a real scenario it is likely that support for one or more of the symptoms will be less than [1.0 1.0]. For example, some of the symptoms may be completely uncertain due to lack of appropriate measurements. In this case, the first measure will yield a [0.0 1.0] support which does not reflect the evidence of the non [0.0 1.0] symptoms.

Measures not using the [0.0 1.0] symptoms will give more realistic supports. The measure giving the maximum necessary support is used.

Having been discussed above in general terms, the representation can now be discussed in detail. A notation will first be defined, and then a method of inference described. The section will be concluded with an extensive worked example.

Notation of representation.

Let P be a fault.

1) $D = \{d_1, d_2, d_3, \ldots\ldots, d_n\}$ denotes a definitive description of the fault P, comprising a set of complementary descriptions di (i = 1 .. n). Each description di denotes some behavioural characteristic(s) of the machine, associated with the fault. For example:

na_crack (*speed*)

denotes the change in the *speed* aspect of machine behaviour, arising from the fault P.

The set $\{d_i\}$ as a whole, represents a change in machine behaviour which characterises the fault P. If this characteristic behaviour is observed then it is reasoned that the fault exists.

In practice, there will be aspects of behaviour that cannot easily be represented or measured. Therefore a reasonable approximation to the definitive description is required.

The (theoretical or ideal) definitive description is denoted by (D_{exact}), and its reasonable approximation by D.

Since D is not exact, the maximum necessary support that can derive from it, will be typically less than 1.0. This number is denoted by Dl.

The above ideas can be loosely expressed using support logic notation, as:

P :- behaviour characterised by D : [Dl, 1.0].

An upper conditional support of 1.0 ensures the rule is used only to provide evidence in support of the fault. Negative evidence is assessed using the rules described in an earlier section.

2) m_{ij} denotes a particular measurement of d_i, i.e. a particular conjunction of symptoms giving a measure of support for the behaviour characterised by d_i. Each j specifies a particular conjunction of symptoms.

The conditional necessary support for d_i given m_{ij} is denoted by ml_{ij}. This may be loosely represented in support logic notation as:

behaviour characterised by d_i :- m_{ij} : [ml_{ij}, 1.0].

An upper conditional support of 1.0 ensures the rule is used only in support of the behaviour.

3) ol_{ij} denotes the necessary support for mij. This necessary support is product of the necessary supports for the conjuncts of mij. The necessary supports for the conjuncts of m_{ij} derive either directly or indirectly (via other rules) from observations of machine behaviour.

4) tl_i denotes the threshold support of the machine behaviour denoted by d_i. It is the maximum necessary support that can be assigned to P without considering d_i. Thus, the threshold value tl_i is independent of the choice of definitive description D.

An alternative description of tl_{ij} is that it is the necessary support for D_{exact} being a definitive description of P, given that it (D) considers all evidence other than the behaviour d_i.

Acquisition of the above knowledge is discussed in chapter 5. An interactive method is proposed, in which the expert enters knowledge to the knowledge base by answering a series of simple questions. An implementation of this scheme is discussed in chapter 6.

Representation and method of inference.

A representation and method of inference will now be discussed using the above notation. The underlying principals are very simple and intuitive. However, their description and implementation appear somewhat more complex.

The scheme attempts to represent knowledge necessary to positive hypothesis testing, using a minimum number of "rules", but retaining the expert's flexibility.

Knowledge is used to make best use of available evidence, however great its uncertainty. Even if many relevant symptoms have [0.0 1.0] supports, the necessary support for the fault should be a reasonable reflection of the remaining evidence.

A further important feature of the scheme is that it facilitates knowledge acquisition, and explanation of diagnoses.

Since the scheme has an intuitive basis knowledge can be collected from an expert by his answers to a few simple questions. These questions can be planned in advance, and will be similar for each fault or application.

Explanation relies largely on the operator's understanding of the *language* used by the system. If the internal representation of knowledge and inference method are intuitive, then they will be more easily related to the operator.

The current implementation of overall hypothesis testing first carries out negative testing. If the resulting possible support for the fault is lower than some datum, then the hypothesis is assumed to be refuted. Only if the possible support is greater than the datum, is positive testing carried out.

In the latter case, there are two support pairs to be combined, one from the negative testing and one from the positive testing. The method of combination used is discussed later in this section.

Theory of representation and inference:

The final support pair [Pl, Pu] assigned to P must reflect:

A) The range of behavioural characteristics used as a basis for description of P, (and therefore the range of evidence used to assess the evidence). The evidence considered is represented by D. Its effectiveness in testing P is represented by Dl.

B) The choice of measures used to assess the aspects of behaviour d_i (i = 1 to n). The permissible measures for each d_i are m_{ij} (j = 1 to m). The effectiveness of m_{ij} in measuring the evidence for d_i is represented by ml_{ij}.

C) The machine's behaviour. This is used to derive supports for the (symptoms) conjuncts of the m_{ij}. These are represented by ol_{ij}. Thus:

m_{ij} :- **machine behaviour**: [ol_{ij} ,ou_{ij}].

The possible support ou_{ij} is not important to the representation since only supportive evidence is being considers.

D) The relative importance of the behavioural characteristics d_i of P. This is represented by the values tl_i (i = 1 to n)

Let [NSP 1.0] be the positive support for P, deriving from the behaviour d_i (i = 1 to n), i.e :

P : [NSP 1.0]

This support will result from evaluation of a *support logic* rule of the form:

P :- {S} : [CNSP 1.0]

where {S} a conjunction of symptoms. The choice of symptoms is described in detail in the following. Briefly, measures m_{ij} are chosen for each d_i, so that the above rule gives maximum necessary support NSP. {S} is a conjunction of the symptoms comprising the chosen m_{ij} (i = 1 to n).

CNSP is the conditional support for P given by {S}. It depends on the values Dl, ml_{ij}, and tl_i (i = 1 to n).

Central to assessment of the support is the choice of mij for each i. The criterion for choice of each m_{ij} is maximisation of the resulting necessary support NSP for P.

Consider the dependence of CNSP on the values Dl, ml_{ij}, and tl_i:

X) from (A) above it clear that CNSP <= Dl (since the conditional support for P must reflect the range of behaviour considered in support of it).

Y) CNSP will reflect the effectiveness of the measures m_{ij} in measuring the machine behaviour d_i (i = 1 to n). CNSP will increase with increasing ml_{ij}.

From (D) above, if $ml_{ij} = 0$, then :

$$CNSP <= tl_{ij}$$

independent of the composition of D.

One expression satisfying the above constraints and dependences is:

$$NSP = Dl * PROD^{i=1..n} CNSP_i$$

where:

$$CNSP_i = tl_i + (1 - tl_i) * ml_{ij}.$$

The selections of each m_{ij} for each d_i (i.e. the choices of j) are made, to maximise the resulting value of NSP.

The result of evaluating:

P :- {S} : [CNSP, 1.0].

is given by:

$$NSP = MAX^{j} \{ \{ PROD^{i=1..n} ol_{ij} \} * CNSP \}.$$

where:

$$CNSP = Dl * PROD^{i=1..n} \{tl_i + (1-tl_i) * ml_{ij}\},$$

and:

MAX^j denotes maximisation with respect to j for each i.

Therefore:

$$NSP = Dl * MAX^{j} \{ PROD^{i=1..n} \{tl_i + (1-tl_i) * ml_{ij}\} * ol_{ij}\}.$$

Maximisation of NSP can be achieved by maximising each of the multiplicands:

$$\{ tl_{ij} + (1-tl_{ij}) * ml_{ij} \} * ol_{ij}$$

with respect to choice of j.

The end result is a conjunction of symptoms {S} comprising the conjuncts of m_{ij} (i=1 to n), where j is chosen to maximise the above.

Evaluating the rule:

P :- {S} : [CNSP 1.0]

should now assign realistic support for P.

An illustrative example of positive hypothesis testing.

P = non-axisymmetric rotor crack = na_crack

1) A reasonable definitive description D is:

D = {d1, d2, d3 d4},

where d_i (i = 1 to 4) are the aspects of behaviour:

na_crack (*speed*),
na_crack (*locational*),
na_crack (*external*),
na_crack (*temporal*).

The support for D above being definitive (i.e. being D_{exact}) is say:

D = 1.0,

so that:

trotcrack :- **behaviour characterised by D** : [1.0, 1.0].

2a) For the aspect of behaviour d1 = **na_crack** (*speed*):

Symptoms used to measure d1 are:

ssmcs, sfsr, and **ccs,**

therefore possible measures are:

$\underline{m_{1j}}$:- {*speed* symptoms} : $\underline{[ml_{1j}, 1.0]}$.

m_{11}	:-	**ccs,ssmcs, sfsr**	: [1.00, 1.00].
m_{12}	:-	**ssmcs, sfsr**	: [0.90, 1.00].
m_{13}	:-	**ccs, sfsr**	: [0.30, 1.00].
m_{14}	:-	**sfsr**	: [0.20, 1.00].
m_{15}	:-	**ccs, ssmcs**	: [0.70, 1.00].
m_{16}	:-	**ssmcs**	: [0.50, 1.00].
m_{17}	:-	**ccs**	: [0.15, 1.00].
m_{18}	:-		: [0.00, 1.00].

2b) For the description d2 = **na_crack** (*locational*):

Symptoms used to measure d2 are:

brgseveral,

therefore possible measures are:

$\underline{m_{2j}}$:- {*location* symptoms} : $\underline{[ml_{2j}, 1.00]}$.

m_{21}	:-	**brgseveral**	: [1.00, 1.00].
m_{22}	:-		: [0.00, 1.00].

2c) For the description d3 = **na_crack** (*external*)

Symptoms used to measure d3 are:

rc2,

therefore possible measures are:

m_{3j} :- {*external* symptoms} : $[ml_{3j}, 1.00]$.

m_{31} :- **rc2** : [1.00, 1.00].
m_{32} :- : [0.00, 1.00].

2d) For the description d4 = **na_crack** (*temporal*):

Symptoms used to measure d4 are:

prog, dwmtemp,

therefore possible measures are:

m_{4j} :- {*temporal* symptoms} : $[ml_{4j}, 1.00]$.

m_{41} :- **prog, dwmtemp** : [1.00, 1.00].
m_{42} :- **dwmtemp** : [0.30, 1.00].
m_{43} :- **prog** : [0.30, 1.00].
m_{44} :- : [0.00, 1.00].

3) The thresholds for the d_i are:

aspect of behaviour	threshold.
na_crack(*speed*)	0.10
na_crack(*locational*)	0.20
na_crack(*external*)	0.80
na_crack(*temporal*)	0.50

The above shows that the order of importance of the four aspects of behaviour is (most important first): *speed*, *locational*, *temporal*, *external*.

4) Given observed values:

ccs :[1.00, 1.00].

ssmcs :[1.00, 1.00].

sfsr :[0.75, 0.75].

rc2 :[0.00, 0.00].

brgseveral :[1.00, 1.00].

prog :[1.00, 1.00].

dwmtemp :[1.00, 1.00].

the computed values are:

i	j	ol_{ij}	$\{tl_i+(1-tl_i)*ml_{ij}\}*ol_{ij}$	$=A_i$	nPl_i
1	1	0.75	{0.1+(1-0.1)*1.00}*0.75	=0.75	1.00
1	2	0.75	{0.1+(1-0.1)*0.90}*0.75	=0.68	
1	3	0.75	{0.1+(1-1.0)*0.30}*0.75	=0.28	
1	4	0.75	{0.1+(1-0.1)*0.20}*0.75	=0.21	
1	5	1.00	{0.1+(1-0.1)*0.70}*1.00	=0.73	
1	6	1.00	{0.1+(1-0.1)*0.50}*1.00	=0.55	
1	7	1.00	{0.1+(1-0.1)*0.15}*1.00	=0.24	
1	8	1.00	{0.1+(1-0.1)*0.00}*1.00	=0.10	
2	1	1.00	{0.2+(1-0.2)*1.00}*1.00	=1.00	1.00
2	2	1.00	{0.2+(1-0.2)*0.00}*1.00	=0.20	
3	1	0.00	{0.8+(1-0.8)*1.00}*0.00	=0.00	
3	2	1.00	{0.8+(1-0.8)*0.00}*1.00	=0.80	0.80
4	1	1.00	{0.5+(1-0.5)*1.00)*1.00	=1.00	1.00
4	2	1.00	{0.5+(1-0.5)*0.30}*1.00	=0.65	
4	3	1.00	{0.5+(1-0.5)*0.30}*1.00	=0.65	
4	4	1.00	{0.5+(1-0.5)*0.00}*1.00	=0.00	

5) The rules selected to measure the aspects of behaviour d_i of **na_crack** are those underlined above, i.e. :

m_{ij} :-	{symptoms}	:[ml_{ij}, 1].	$_{n}Pl_{i}$
m_{11} :-	**ccs, ssmcs, sfsr**	:[1.0, 1].	**1.00**
m_{21} :-	**brgseveral**	:[1.0, 1].	**1.00**
m_{32} :-		:[0.0, 1].	**0.80**
m_{41} :-	**prog, dwmtemp**	:[1.0, 1].	**1.00**

6) The resulting rule for positive diagnosis of na_crack is:

na_crack :- {S} : [CNSP, 1.0].

where:

{S} = { m_{11}, m_{21}, m_{32}, m_{41} }

= { **ccs, sscms, sfsr, brgseveral, prog, dwmtemp** }

CNSP = Dl * { $PROD^{i=1..n}$ $CNSP_i$ }

= 1.0 * { 1.0 * 1.0 * 0.8 * 1.0 }

= **0.80**

7) The support for the conjunction {S} is:

NEC {S} = ol_{11} * ol_{21} * ol_{32} * ol_{41}

= (1.0*1.0*0.75) * (1.0) * (1.0) * (1.0*1.0)

= 0.75 * 1.00 * 1.00 * 1.00

= **0.75**

POS {S} = NEC {S} = 0.75

8) The final operation is to evaluation the rule:

na_crack :- ccs,ssmcs,sfsr,brgseveral,prog,dwmtemp : [0.8,1]

Since:

{ccs,ssmcs,sfsr,brgseveral,prog,dwmtemp} : [0.75,0.75]

the final support for **na_crack** is:

na_crack : [0.60, 1.0].

The necessary support is 0.6 (less than 1.0) because two relevant symptoms have poor supports:

rc2 : [0.00, 0.00], and

sfsr : [0.75, 0.75].

4.3.5 The overall method of hypothesis testing.

The overall method of hypothesis testing combines the above explained negative and positive approaches.

Negative testing applies evidence in such a way as to attempt to show that faults do not exist.

Positive testing applies evidence in such a way as to attempt to show that faults do exist.

Negative testing is performed first.

If negative testing is successful, positive testing is not performed.

If negative testing is not successful then positive testing is performed and the two support pairs combined.

Support pairs derived from the two types of testing are of the form:

negative testing :

P : [0.0 PSP]

positive testing :

P : [NSP 1.0].

PSP represents 1.0 minus the evidence supporting negation of the hypothesis that fault P exists.

NSP represents the evidence supporting the hypothesis that fault P exists.

If PSP is lower than a predefined value, evidence against the hypothesis is assumed to outweigh likely evidence for it. A reasonable cut-off point would be, say 0.1.

If PSP is below the 0.1 cut-off, the overall support pair is assumed to be:

P : [0.0 PSP]

If PSP is above or equal to 0.1, positive testing is performed, and the overall support pair given by:

P : [NSP1 PSP1],

which results from the Shafer-Dempster combination of:

P : [NSP 1.0]

and

P : [0.0 PSP].

4.4 Choice of remedial actions.

Remedial actions are the means of eliminating deviation of machine states from their ideal. They are selected using a cost model.

Possible remedial actions are listed below:

r0) no action,
r1) increased vibration or other measurement,
r2) exploratory test(s), eg deloading the machine to see the effect it has,
r3) implementing operational restriction(s), eg running the machine at reduced load,
r4) take the machine out of service, in order to perform exploratory work or essential maintenance.

Choice of remedial action(s) aims to minimise the overall cost associated with a fault. This cost comprises:

e1) cost associated with the machine's loss of function, and
e2) the cost of performing remedial action(s).

Cost minimisation is complicated by a number of factors:

u1) uncertainty about which faults actually exist,
u2) inability to quantify costs associated with loss of functionality,
u3) uncertainty about likely success of remedial action(s).

Notation:

F_i (i=1..n) are faults , and

F_i : $[Fl_i, Fu_i]$ represents evidence supporting the faults.

$CF_i = [CFl_i, CFl_i]$ represents the anticipated cost of fault F_i. The range of possible values for the cost is CFl_i to CFu_i.

R_j (j=1 to m) are remedial actions, and

R_j : $[Rl_{ij}, R_{uij}]$ represents evidence that R_j is a remedy for fault F_i.

$CR_j = [CRl_j, CRu_j]$ represents the cost of performing remedial action CR_j. The range of possible values for the cost is CRl_j to CRu_j.

$U_j = [Ul_j, Uu_j]$ represents the range of possible values for the utility of remedial action R_j.

$B_j = [Bl_j, Bu_j]$ represents the benefit derived from performing remedial action R_j.

The utility of a remedial action will be defined as:

utility of R_j = benefit derived from R_j - cost of R_j,

or:

$$U_j = B_j - CR_j. \qquad (1).$$

From conventional probability theory, the expected value of an event whose success provides X pounds and whose probability of success is P would be given by:

$$E = X * P$$

This similarly applies to remedial actions. They have a probability of success and associated "payouts", although neither are known exactly (but ranges for them can be

assumed). If X and P above were replaced by ranges [Xl,Xu] and [Pl,Pu], the equation would also be replaced:

$[El, Eu] = [Xl*Pl, Xu*Pu]$.

It follows from above that the expected benefit B_j can be expressed in terms of:

1) evidence of possible faults,

2) costs associated with possible faults, and

3) evidence that remedial actions R_j will cure faults that exist.

$$B_j = [Bl_j, Bu_j]$$

where:

$$Bl_j = \sum^{i=1..n} (Fl_i * Rl_{ij}) * CFl_i. \qquad (2.1)$$

$$Bu_j = \sum^{i=1 \quad n} (Fu_i * Ru_{ij}) * CFu_i. \qquad (2.2)$$

$[Fl_i*Rl_{ij}, Fu_i*Rl_{ij}]$ is the evidence that remedial action R_j will alleviate fault F_i. This is derived from evidence $[Fl_i,Fu_i]$ of fault F_i, and evidence $[Rl_{ij}, Ru_{ij}]$ that remedial action R_j will cure F_i.

Substituting (2.1) and (2.2) into (1) gives:

$$Ul_j = \sum^{i=1..n} (Rl_{ij} * CFl_i) * Fl_i - CRl_j. \qquad (3.1)$$

$$Uu_j = \sum^{i=1..n} (Ru_{ij} * CFu_i) * Fu_i - CRu_j. \qquad (3.2)$$

This expresses the overall utility of a remedial action, in terms of the evidence which support faults F_i, evidence that remedial actions R_j will cure faults F_i, costs associated with faults F_i, and costs of performing remedial actions R_j. The evidence relating to the faults F_i is determined by diagnosis (described earlier), whilst all other evidence must be known apriori.

The overall utility is maximised by choice of remedial action R_j.

Difficulties may arise when performing the above maximisation, since the utility is expressed as a range rather than a single number. However, it should be possible to achieve at least a partial ordering of the pairs $[Ul_j,Uu_j]$. If necessary the utilities could be ordered by their mid-range values $(Ul_j-Uu_j)/2$.

$[Rl_{ij}, Ru_{ij}]$ representing evidence that remedial action R_j is a cure for fault F_i, can be collected directly from an expert. If exact values can not be given, then qualitative values will suffice.

$[Fl_i, Fu_i]$ representing evidence of fault F_i, is calculated during diagnosis, and therefore known at this stage.

$[CRl_j, CRu_j]$ representing the cost of performing remedial action R_j, can be collected similarly to $[Rl_{ij},Ru_{ij}]$. Costs will typically be associated with maintenance and loss of production.

$[CFl_i, CFu_i]$ representing the expected cost of fault F_i, can derive from a combination of sources:

cf1) loss of function, e.g. due to load restrictions imposed to reduce the effects of a fault such as a **thermal bend**,

cf2) increasing machine damage, e.g. due to fault induced vibration,

cf3) possible catastrophic failure, (perhaps resulting in loss of the machine, and death or injury of operators).

Difficulties arise when trying to quantify the costs of catastrophic failure. Injuries, lives lost, and damage to public opinion, are not easily amenable to financial evaluation.

Despite the obvious difficulties, decisions involving conflict between human and financial worth, are made in many areas (health care and military action, for

example.) This hints at the existence of at least a fuzzy relationship between values of animate and inanimate objects.

In order to produce a simple but realistic cost model of faults, it may be assumed that the cost of catastrophic failure is of the order of the value of the machine. This will strongly bias costs in favour of potentially catastrophic faults, and so ensure a safe policy.

Collection of additional data following inconclusive diagnosis.

If remedial action selection proves inconclusive, additional evidence and therefore machine observation, will be required.

Of the symptoms available for diagnosis some will be uncertain; with support pairs of the form [l,u], indicating uncertainty of magnitude (u-l).

As a result of this uncertainty, there will be uncertainty about hypotheses. Uncertainty about hypotheses may therefore be reduced by reduction in uncertainty about the symptoms.

Uncertainty associated with hypothetical faults will effect the calculated utilities U_j of the remedial actions. This may result in a choice of non-ideal remedial action. There is therefore an expected cost associated with uncertainty about symptoms.

The elimination of uncertainty in the support for a symptom S_k may or may not effect the choice of remedial action. If it does not, then the utility of the reduced uncertainty is zero, if it does then the utility is ${}_{DELTA}U^k$ where:

$$ {}_{DELTA}U^k = U^{k+}{}_{j=s} - U^{k-}{}_{j=t}. \qquad \text{----(DELTA U)} $$

where:

1) $U^{k+}{}_j$ denotes the utility of remedial action R_j after uncertainty about S_k has been reduced,

2) $U^{k-}{}_j$ denotes the utility of remedial action R_j before uncertainty about S_k has been reduced,

3) j = s denotes the best choice of remedial action R_j with reduced uncertainty about S^k,

4) j = t denotes the best choice of remedial action R_j without reduced uncertainty about S_k.

It is assumed that the updated support $[l^+,u^+]$ for S_k, obtained after extended observation, will be such that:

$$(u^+ - l^+) <= (u^- - l^-).$$

It is further assumed (to simplify matters) that $_{DELTA}U^k$ will be maximised with respect to S_k in one of two ways:

$$(l^+, u^+) = (u^-, u^-)$$

or

$$(l^+, u^+) = (l^-, l^-).$$

This will maximise $DELTAU^k$ with respect to reduction in uncertainty about the symptom S_k.

Utility associated with uncertainty about support for S^k is calculated as follows (the uncertain support for S_k is [l,u]]):

1) Fault diagnosis is carried out, and the utilities of each remedial action computed; the maximum is denoted $U^{k-}_{j=t}$.

2a) The support [l,u] for symptom S_k, is replaced by [l,l] and step (1) repeated. This gives a new set of utilities, the maximum being denoted $U^{k+}_{j=s1}$.

2b) The support [l,u] for symptom S_k is replaced by [u,u] and step (1) repeated. This gives a new set of utilities, the maximum being $U^{k+}_{j=s2}$,

2c) The maximum of $U^{k+}_{j=s1}$ and $U^{k+}_{j=s2}$ is denoted $U^{k+}_{j=s}$,

3) If t = s, removal of uncertainty about S^k will not effect the choice of remedial action, and the utility of reduction in uncertainty about S^k is [0, 0].

4) if t <> s (t not equal to s), the maximum utility of reduction in uncertainty about S^k is:

$[0, (U^{k+}_{j=s} - U^{k+}_{j=t})]$.

The necessary utility is zero because the reduction in uncertainty about S_k cannot be foreseen. The above is the utility of R_j corresponding to bounds for the updated support for S^k.

The possible utility, is simply the difference between utilities of what might possibly be the optimal remedial action, and what is currently thought to be optimal. In other words, because of uncertainty about the support for S_k, remedial action might be incorrectly selected.

Given further observation, uncertainty about S_k could be reduced so as to indicate alternative optimal remedial action. The new support for S^k (with lower uncertainty) will provide more realistic utilities U^k.

Recap:

The first step following an inconclusive diagnosis, is to examine evidence on which the diagnosis is based, and consider the possible merit of further observation. This may for example, simply involve more detailed examination of existing measurements. With reduced uncertainty about symptoms it is possible to make more reliable choice of remedial action.

The utilities associated with uncertainty in symptoms, indicate the order in which symptoms should best be collected.

The overall strategy can be summarised:

1) obtain supports for relevant symptoms,

2) assess support for possible causes (faults and operating conditions),

3) where there is conclusive diagnosis, select appropriate remedial action,

4) where there is inconclusive diagnosis, obtain additional support for symptoms, and repeat from 2).

CHAPTER 5

KNOWLEDGE ACQUISITION

5.1 Overview.

Knowledge acquisition is the process by which expertise is transferred to an expert system.

There are essentially two approaches to knowledge acquisition. It may be performed either 'automatically' by the expert system itself or 'manually' by a knowledge engineer.

5.2 Automatic knowledge acquisition.

Automatic knowledge acquisition (AKA) is performed by a component of the expert system shell, whose purpose is to induce expert knowledge from 'a learning set'.

A learning set is a collection of examples consisting of problems and associated solutions. The AKA module finds a knowledge base which 'explains' (using a set of rules for example) how solutions are associated with problems in the learning set. The aim is to generalise the induced rules and then to use them to solve future problems.

There are similarities between this approach and scientific method. Existing evidence gives rise to a set of beliefs which enable predictions about how 'the world' will behave in the future. Belief in induced knowledge is reinforced whenever it helps to produce correct predictions, but is brought in to question when it helps to produce false predictions. The overall structure of the induced knowledge sustains until there are a significant number of false predictions. At this point the original 'learning set' is augmented with the unsolved problems and their 'real solutions' and a new and hopefully better knowledge base is derived.

The method differs from scientific method since the AKA can not perform experiments to test the hypotheses which it generates.

Consider the problem of fault diagnosis for turbo-generators. The aim is to infer faults on the basis of available information about machine behaviour. The AKA module uses a learning set comprising (to some degree) confirmed faults and the sets of symptoms that where exhibited by the machine at that time.

The combined set of instances of machine behaviour forms a 'behaviour space' within which (it is hoped) there are certain areas which can be associated with specific faults. Future suspect machine behaviour is assigned a position in the behaviour space and the offending fault or condition thereby determined.

This method is described intuitively as pattern classification. By assigning a new point to the space its data set is effectively classified by a label given to the area of space in which it is placed.

There are various methods of implementing pattern classification. The most obvious and familiar method is to represent knowledge implicit in the 'behaviour space', explicitly, as a set of IF...THEN rules using (for example) PROLOG (or FRIL its evidential counterpart).

It is unfortunate that, to induce knowledge accurately and with sufficient statistical confidences, large 'example sets' are necessary. The number of examples available to the author of this work was very limited, and in any case it was felt that the CEGB did not have sufficient numbers of complete examples to enable knowledge induction.

Many of the 'fault reports' in existence were, for example, incomplete. Some are lacking documentation of either the fault or some important symptoms. During discussions with experts a feeling emerged that many diagnoses were based on 'intuition'. This could mean:

1) the experts used data or knowledge that they were unable to quantify (and therefore unable to document or divulge), or

2) the experts did not understand their own reasoning sufficiently well to give an adequate account of it.

A further barrier to obtaining a reasonable number of 'fault reports' appeared to be the CEGB's own security regulations regarding control of restricted or 'sensitive' information. In the second and final years of the project the original industrial supervisor was relocated and given different responsibilities. Subsequent contacts at the CEGB appeared less willing to release documents. Presumably they felt that if security regulations were strictly applied they could be found in breach of security.

Since, in these circumstances, automatic knowledge induction was unfeasible, a engineering approach to knowledge acquisition was taken.

5.3 An engineering approach to knowledge acquisition.

The engineering approach to acquiring knowledge for expert systems relies on the work of a knowledge engineer.

The knowledge engineer elicits knowledge and engineers a knowledge base using one or more of the following sources of knowledge:

1) case histories,
2) experts,
3) system models.

Each of these are discussed in the following sections.

5.3.1 Knowledge acquisition from case histories augmented by expert consultation.

A case history comprises details of suspect machine behaviour and resulting conclusions drawn by the experts involved. Some level of explanation of the underlying reasoning will most likely be given.

Details of the reasoning involved in the process are most important from the point of knowledge acquisition.

Knowledge used in a diagnosis (for example) will fall into one of three categories:

1) knowledge used is explicitly stated in the accompanying explanation, and is therefore easily extracted from the text. However, care should be taken to determine the 'scope' of applicability of the knowledge,

2) knowledge used is implicit in the explanation and can be induced from it (some examples are given later in this chapter),

3) knowledge used is not explicitly stated and cannot be unambiguously induced. In this case it may be possible to make an educated guess at it and/or consult with an expert.

It is important that the knowledge engineer determines the scope (or range of applicability) of any knowledge that is acquired. In fact, it is not properly acquired until this is done.

A primary objective of knowledge acquisition is to generalise acquired knowledge to the highest possible level, but no further. For example, a piece of knowledge which is induced from a single case history will have been used in relation to a particular machine. The knowledge may therefore only be applicable to this one machine, or it may be more generally applicable, for example to all machines, or all machines of a certain design.

It is unusual to find the scope of knowledge (described in an explanation) to be stated, since 'fault reports' are not necessarily intended to inform their readership about fault diagnosis. The aim is more commonly to record events and circumstances surrounding a case (some of the symptoms for example), and to justify diagnoses to superiors. The latter aim might tend to force conflicting evidence out of a report or reduce its prominence within the report.

In some cases, the scope of obtained knowledge will be deduced by the knowledge engineer using existing or common sense knowledge. For example, Newtonian mechanics confirms the almost universal scope of the following piece of knowledge:

"A chance in the balance of a rotating shaft results in a change in the phase and amplitude of the shaft's vibrations at the once-per-revolution frequency."

For this and other reasons, it is important to validate any knowledge collected from case studies. The only feasible methods of achieving this are corroboration by other written material or by expert consultation.

Expert consultation is the most efficient method of clarifying specific uncertainties about expert knowledge. Ambiguities can usually be cleared up quite quickly with an expert's help unless the uncertainties are shared by the expert. The same clarification might only be achieved after hours of sifting through written material.

The disadvantage of expert consultation is that experts are often too busy to be readily available. Bearing this in mind, the overall approach of the knowledge engineer in acquiring the bulk of the knowledge could be:

1) use expert consultation to discern the overall structure of knowledge relevant to the subject area,

2) obtain as much knowledge from case studies as is reasonable (ie. until a point of diminishing returns),

3) use expert consultation to validate and augment knowledge obtained in 2.

Stages 2) and 3) would tend to be used in that order, but could be iterated, in the sense that 1) and 2) would be carried out simultaneously on different areas of knowledge. This enables the knowledge engineer to avoid bottle-necks which occur when access to the expert is necessary but not possible.

Expert consultation is also recommended in the latter stages of knowledge acquisition, when a representation or framework for the knowledge has been devised.

Experience has shown that the following three broad guidelines should be observed (in the latter stages at least):

1) at each consultation the knowledge engineer should have clear and reasonable objectives for each consultation session,

2) questions directed to the expert should be clearly stated preferably using mutually agreed terminology, and contain a minimum of ambiguity,

3) questions posed to the expert should be answerable in simple terms and preferably by selection from a predefined set. For example, 'is the answer x,y, or z ?'. This should ensure that the expert's replies are 'to the point' and intelligible to the knowledge engineer.

5.3.2 Knowledge acquisition using expert consultation.

Expert consultation has already been discussed above, as an aid and supplement to knowledge acquisition from case studies. Some more general points are now addressed.

The first point relates to the initial stages of knowledge acquisition during which the knowledge engineer is new to the domain, and the expert is most likely unused to discussing his expertise.

Since the interviewee is an expert, he will inevitably use specialist terminology. So, an important first step is for the knowledge engineer (aided by the domain expert) to define the terminology that is likely to be used in the subsequent sessions. This is effectively the definition of a common language to enable communication.

An important part of the process is to clearly define the problem in terms of the possible faults, remedies, and symptoms (for example).

Symptoms are a major input to condition monitoring and must therefore be properly described. Description of symptoms should include:

1) means of collection, and hence the ease with which they can be updated and corroborated,

2) frequency of measurement, and hence their likely availability at the outset of a diagnosis or on request,

3) reliability,

4) whether they are deterministic or non-deterministic (ie. whether they have associated uncertainty or fuzziness).

Having first agreed a set of terminology (and thereby restricted the scope of the problem), a suitable method of knowledge representation must be chosen or invented. It is likely that any suitable 'off the peg' representations will be made obvious by the initial 'definitive' stage. If one does emerge then it may as well be used, but if no 'off the peg' representation is adequate one may have to be invented or an existing one adapted.

Extended, general discussion between the expert and the knowledge engineer (combined with the reading of case studies) may be needed before making a final choice of representation. In this case, discussion with the expert should be open ended and the full scope of the problem carefully considered. This will help to prevent the problem being forced into a representation that appears 'reasonable' at the time but later proves to be 'unreasonable'.

Initial discussions will also inevitably contain an element of open ended questioning for example, 'tell me how you diagnose faults', as opposed to the more formalised approach used in latter stages of knowledge acquisition.

The second general point about expert consultation to be remembered, is that the expert's perception of how he performs his task is not necessarily accurate. For example, if someone is asked to explain how they recognise a certain person they will (most likely) be able to give an opinion as to the reasons. However the reasons given will be incomplete, possibly incorrect, and with the wrong emphases.

The fact is that 'face recognition' is a very complex procedure and no one can give an adequate natural language explanation of how they do it.

Except for trivial examples, the recognition process just is not amenable to the simple piecewise description arising from natural language explanation.

The above 'recognition' principal applies similarly to turbo-generator state monitoring although the problem is less complex and much of the reasoning involved is

performed (at least consciously) at a higher level. Some of this reasoning can be explained qualitatively and is amenable to a form of logical representation.

The lower level reasoning is less easy to explain and represent logically. For example, how can the differences between two graphs be explained taking in to account the relative importance of different parts.

There is an another potential problem in that there is evidence that experts deliberately (although often unconsciously) omit vital pieces of knowledge and/or over-complicate their task. An expert might not feel very flattered at the prospect of having much of his hard earned experience represented in a few hundred or so rules, possibly (he may feel) at the expense of his livelihood.

5.3.3 Knowledge acquisition tools.

The knowledge engineer can sometimes use knowledge acquisition tools to assist and expedite knowledge acquisition whether from case histories (examples) of from an expert (or both). Two such tools are 'Repertory grid analysis' and 'Multi-dimensional scaling'.

Repertory grid analysis is a technique for determining the way in which an individual perceives certain objects (items or events).

Kelly's 'Psychology of Personal Constructs' [Kelly, 1955] asserts that a persons processes are psychologically channelised by the way they anticipate events and that events are anticipated by describing them in terms of personal 'psychological constructs'. More simply, each person has a set of descriptions that he uses to understand objects. Each object or event is understood using a number of appropriately weighted descriptions.

This gives the view that each individual is a scientist. The aim in construing an object in a certain way (theorising) is to predict the future involving that object. If future events show the construction to be false (refutation) the object is reconstrued (the theory revised).

Repertory grids are used to determine the way in which certain objects are construed ie. which constructs are used and with what weighting. These are described in [Bannister & Fransella, 1971].

There are essentially two ways of implementing the method to examine a set of objects.

The simplest method is firstly to determine the relevant constructs. For Each object, the expert is asked to assign a weighting to reflect the applicability of each construct. The expert is given a list of permissible weightings from which he may select one. The weightings typically range from: 'not at all applicable' to 'very applicable', with perhaps three intermediate levels.

Another (more rigourous) way of implementing repertory grid analysis of a set of objects is to consider them three at a time (using suitable combinations) to determine the constructs and their weights. This approach dispenses with the need to explicitly label constructs at the outset. The constructs emerge in the course of the procedure.

The method proceeds by considering the objects in combinations of three. For each group of three the expert indicates which two are most alike and which two are least alike (and to what extent). The system then invents constructs (ie. names) and weights to explain the differences and similarities between the objects. At this point, the constructs are only implicitly labelled by virtue of the way they describe the objects. Explicit labels can be assigned to the constructs by the expert if necessary.

There are obvious uses of repertory grid analysis to the acquisition of knowledge about turbo-generator fault diagnosis. For example, the above described method could be applied to the set of possible machine faults. This would determine the important machine conditions (or symptoms) and their relationship to machine faults (and thereby the means of fault diagnosis).

Multi Dimensional Scaling is a set of techniques that allows data objects to be graphically represented so as to highlight similarities and dissimilarities between them.

Each data object is represented by a point in a space of specified dimensions (the following assumes the dimension to be two) where the distance between any two points represents their dissimilarity.

The method is illustrated by the following. Consider the problem of acquiring fault diagnostic knowledge from an expert.

Faults are diagnosed on the basis certain symptoms. To see which symptoms are relevant to which faults it is necessary to analyse a large number of examples.

Each example comprises a set of symptoms and the fault causing them. The symptoms are treated as the independent variables and the type of fault as the response variable. For each fault the values of independent variables form a data set (each data set used the same set of variables although only a subset of them will be relevant to a particular fault.

The MDS system firstly represents the independent data sets as a map of points in space. This is represented graphically as a polar type plot. Points that are close together represent data sets that the system believes to be closely related, and conversely.

The next step in the process is performed by the knowledge engineer. Each point on the map is marked in some way to show the type of fault which caused them (ie. the value of the dependent variable). This will hopefully show a pattern in which similar faults tend to be grouped in localised areas of the space.

The final stage in the process is performed by the MDS system. The data sets are reexamined and another polar type plot produced. This shows the effect of the independent variables on the distribution of the map of points produced in stage 1 of the process.

The method shows which independent variables (or symptom) are associated with which faults and to what extent.

The confidence that can be given to the results is largely dependent on the number and range of examples considered.

5.4 knowledge acquisition via system modelling.

System modelling is a useful and important means of knowledge acquisition. The source of knowledge used is a deterministic or semi-deterministic model of the system in question.

This type of model can be used 'as is' to provide answers to or evidence regarding specific questions. For example, deterministic models of the turbo-generator can be used to examine the likely distribution of vibrations along the machine's shaft, and thereby give supporting or opposing evidence to fault hypotheses.

Deterministic models enable the expert systems to give better answers than they could otherwise do. Such models therefore represent knowledge, and constructing the models is effectively knowledge acquisition. It may seem unusual to describe deterministic modelling in this way because knowledge (in AI terms) is more commonly represented explicitly (as predicate logic for example) rather than implicitly. However, it is sometimes possible to produce explicit knowledge from the deterministic models if this is preferable.

Implicit and explicit representations of deterministic models each have their advantages and disadvantages.

Implicit representations usually result in slow processing but enable knowledge to updated more easily (since they are a more natural representation).

Explicit representations usually result in fast processing but are unnatural. To update the knowledge it is therefore necessary to go back to the implicit model and regenerate the explicit 'equivalent'. Also, the explicit representation will necessarily only contain a subset (although possibly a qualitatively sufficient one) of the knowledge contained in the implicit model.

Only limited system models exist for the turbo-generator because of its complex behavioural characteristics.

5.5 Illustrations of knowledge acquisition.

5.5.1 Illustrations of rule induction from case studies.

A few examples of rule induction from fault reports will now be presented. Each example comprises a passage taken from a report, a list of rules induced from it, and a brief explanation of the underlying reasoning.

The faults reports [Allen, 1981; Myers, 1983, 1984; Myers & Thomas, 1982; Nichols, 1984] were all written by experts at the South Western Regional Headquarters of the CEGB.

Passage 1:

On the morning of <date> the machine was running with a steady load of 500 MW, and a steady rotor current if 3650 A. There were no operational events immediately prior to or at the time of the incident. However, the generator was operating with higher than usual but not abnormal hydrogen temperatures, due to a fault on a hydrogen cooler.

Induced rule:

machine in stable state at time <= T AND
machine not in stable state at time > T AND
operational event at or just before T **===>**
it is possible that the cause of the loss of stability was associated with the operational event.

where the symbol "===>" indicates a causal implication.

The above rule follows from passage 1, using induction and generalisation. The deduction contained in the passage makes use of a widely understood principal.

Passage 2:

In spite of the increase in 1/rev vibrations, the 2/rev vibrations were essentially identical to previous run-downs (FIG). It was therefore concluded that a transverse crack was not the change. This also implies that the cause of the change is some sort of balance change on the rotor. This is because of the observed response on the bearings arises from two factors, the exciting force and the response characteristics of the system. The 2/rev vibrations are the response of the system to a 2/rev excitation etc.

Rule 1:

a change in the n/rev excitation force OR
a change in the response characteristics <===>
a change in the response of the system at n/rev.

"<===>" indicates a bi-directional causal relationship between its right and left hand side.

The rule is more or less explicitly stated in the final two sentences of the passage, except for one generalisation (ie. 2/rev has been replaced with n/rev). The reason for the bi-directional implication is that the only possible causes of a change in the response are changes in the response characteristics and/or the excitation force.

Rule 2:

a change in balance ===>
a change in the 1/rev excitation force.

This rule follows from the first and second sentences of the passage. Since the changes are only observed in the 1/rev vibrations, it follows from rule 1 above that only the 1/rev exciting force can have changed. Balance change is identified as the only possible cause. The above rule follows directly.

Rule 3:

a transverse rotor crack ===>
a change in the 2/rev exciting force OR
a change in the response characteristics.

This rule follows from the first and second sentences in a similar manner to rule 2. The passage states that there is no change in the 2/rev response. Therefore, using rule 1 with n equal to 2, it can be deduced that there has been no change in either the response characteristics or the exciting force at 2/rev. These two facts are presented as the basis for eliminating the possibility of a transverse rotor crack and the rule follows.

Note that the above implication is not reversible since it is not known whether or not there are any other possible causes of changes in the 2/rev response.

Further examples of rules induced from case studies:

A rule that is applies to all machines:

a change in the balance of a rotating body ===>
a change in the 1/rev exciting force.

Rules that are applicable to rotor-dynamics:

a change in the stiffness asymmetry of a shaft ===>
a change in the 2/rev exciting force.

loss of mass from the shaft OR
movement of mass on the shaft OR
a bend in the shaft ===>
a change in the balance of the shaft.

a transverse crack OR
a coupling failure ===>
a change in the stiffness asymmetry of the shaft.

5.5.2 Knowledge acquisition in action.

The following example of knowledge acquisition using expert consultancy, embodies the methodology employed in building a knowledge base for turbo-generator fault diagnosis.

The text in bold type is generated by the expert system's knowledge acquisition module to elicit knowledge from the (expert) user.

The methodology behind the knowledge acquisition.

The knowledge representation described in chapter 4, and the knowledge acquisition methodology described in the following are united by a central assumption. The assumption is that an expert's perception of a fault can be viewed at varying levels of abstraction.

For example, at the zero'th (or lowest) level of abstraction a rotor crack is perceived as a discontinuity in the metal of the rotor. This is a graphic and intuitive description of the fault but has very limited use because the rotor is normally shielded by several feet of lagging (and a stator) and is rotating at 50 Hz.

Awareness of the machine is therefore necessarily maintained via indirect observation. This indirection is so extreme that the expert sees only graphical and tabulated vibrational and other data.

At the next level of abstraction (the first, say) a crack could (for example) be defined as:

1) a progressive stiffness asymmetry, and
2) a progressive stiffness change.

These are two complimentary descriptions of a transverse rotor crack and together form a more abstract view of the fault.

The abstraction process can be extended to the n'th degree (or level). At any given level each descriptor can itself be described in terms of other quantities and thereby creates a more abstract description of the fault.

The abstraction process continues until it produces a description that is (in this case) in terms of vibration patterns. This enables the expert to assess machine behaviour represented as vibrational data in the light of his abstract perceptions of possible faults.

The basic idea is that a given fault can be described in terms of the behaviour it produces on the machine. This behaviour is represented by a number of complementary descriptions which derive from the natural grouping of symptoms of the machine's behaviour.

In this work the following groupings of symptoms have been used:

temporal,
locational,
external,
speed,
frequency,
nonvib.

Temporal deals with the variation of behaviour with time.

Locational deals with the spatial variation of behaviour.

External deals with the value of operational parameters that effect machine behaviour.

Speed deals with the variation of machine behaviour with respect to the machine's rotational frequency.

Frequency deals with the variation of vibrational behaviour with respect to rotational frequency.

Nonvib is a 'catchall' that deals with non-vibrational aspects of machine behaviour.

The above groupings (or descriptions) reflect the following facts:

1) that the machine exists in time and space, hence the descriptions *temporal* and *locational*,

2) that the machine is not isolated, and its behaviour therefore depends on outside influence (operational events and parameters), hence the description *external*,

3) that the machines behaviour is a function of its operating frequency, hence the description *speed*,

4) that the vibrational behaviour of the machine can be fourier analysed to reveal important diagnostic information, hence the description *frequency*, and

5) that, although vibrational behaviour is most important in diagnosis, some non-vibrational information about the machine is of use (the bearing metal temperature for example), hence the description *nonvib*.

A short example will now be given to illustrate the working of the expert system's knowledge acquisition module, and to reinforce some of the principals described earlier.

<u>The acquisition of expert knowledge used in the diagnosis of non-axisymmetric cracking.</u>

As explained (at some length) in Chapter 6 (Implementation of the expert system), the expert system's user interface is based around an 'extended menu system'. The expert system can be controlled and supplied with knowledge (and data) by selecting appropriate options from menu's generated by the system.

The menu system displays all permissible instructions that the user can issue and where appropriate (for example where acquiring knowledge) the system displays all similar knowledge so that the expert can maintain overall consistency and give inputs in the correct format.

A detailed description of the design and use of the expert system is presented in chapter 6.

1) The first step in the knowledge acquisition process is to devise a definitive description of the fault using the (earlier mentioned) descriptions:

external,
non-vibrational,
locational,
frequency,
speed,
temporal.

An appropriate subset of the descriptions above is chosen as a definitive description of the fault. For the purpose of this example the subset is assumed to be:

external,
locational,
speed,
temporal.

2) The second step of the process is to determine the extent to which the subset above is truly definitive of the fault. This is (say):

$D_{exact} = 0.8$.

3) The third step of the process is to determine the importance of each of the (complimentary) descriptions comprising the above subset. This is achieved using a question and answer session between the expert and the system. For example:

For the description <desc> of the fault, assign its importance

0 irrelevant
1 not very important
2 fairly important
3 important
4 very important
5 vital.

N = ?

The expert is expected to respond with one of the options 0,1,..,5.

Each possible reply corresponds to a threshold for the description <desc>, with respect to the fault in question (for an explanation of the meaning of *threshold*, consult chapter 4). The relationship of the possible replies (N) to thresholds is as follows:

N	threshold.
0	1.0
1	0.8
2	0.6
3	0.4
4	0.2
5	0.0

If the response is N = 0 then the threshold is 1 and the description <desc> is irrelevant to positive diagnosis of the fault. If the response is N = 5 then the <desc> is essential to positive diagnosis of the fault. N = 5 corresponds to a threshold of 0.0, which (as explained in chapter 4 on knowledge representation and inference) makes it impossible to obtain positive support the fault without using the description <desc>.

4) The fourth step in the process is to find ways to calculate the evidence relating to this aspect of the fault. For example, whether the *temporal* behaviour associated with non-axisymmetric cracking is being exhibited.

Ways of calculating the evidence are defined by a set of measures (also described in Chapter 4) which associate a necessary support with the description.

Again taking the description *temporal* as an example, the procedure is as follows:

4a) Obtain the names of each of the quantities used to measure the necessary support of the description *temporal* of the fault:

Enter the names of all quantities involved in measuring non-axisymmetric cracking (*temporal*):

prog: **changes progressive with time, and**
dwmtemp: **changes in a timescale of days, weeks, & months.**

4b) All possible alternative rules having the specified quantities as conjuncts, are generated by the expert system or the knowledge engineer. The ability of each rule to measure the given description is specified by the expert using a number between 0 and 5 such that :

0 represents useless
1 represents not very useful
2 represents fairly useful
3 represents useful
4 represents very useful
5 represents superb.

For example:

How useful is the rule below

non-axisymmetric cracking (*temporal*) :-

changes progressive with time,

changes in a timescale of days, weeks, & months.

0 useless

1 not very useful

2 fairly useful

3 useful

4 very useful

5 superb

N = ?

Again the expert is expected to respond with one of the above five options 1,2,..,5. Each response (N=) 1,2,..5 corresponds to a conditional necessary support according to the following scheme:

N	conditional necessary support
0	0.0
1	0.2
2	0.4
3	0.6
4	0.8
5	1.0

For example, if the value of N selected is 5 then the corresponding conditional necessary support will be 1.0 and the resulting rule will be:

non-axisymmetric cracking (*temporal*) :-

changes progressive with time,

changes in a timescale of days, weeks, & months,

:[1.0, 1.0].

The above described process is repeated for all possible rules that can be constructed from all quantities (conjuncts) supplied, and for all specified descriptions.

Having obtained all the above knowledge from the expert (relating to positive hypothesis testing), the expert system proceeds to obtain knowledge to enable refutation (ie. negative testing).

It is clear, that if the nonexistence of a symptom can be used to refute a fault, then its presence will corroborate the fault to some extent. Therefore it is reasonable to make an initial assumption that the set of all symptoms cited as being of use in positive diagnosis will contain all the symptoms that may be of use in refutation. Many (if not most) of these symptoms will turn out to be of little importance to the process and will then be disregarded.

If the expert feels that there are symptoms used in negative testing other than those hypothesised by the positive testing, they can be added to the list of important symptoms.

Refutative rules are constructed by first subdividing the set of relevant symptoms in to subsets which then become the rule bodies. In general, the body of a rule will consist of a disjunction of symptoms. For example, a subset of symptoms S1 and S2 will give rise to a rule of the form:

non-axisymmetric cracking :- NOT {S1;S2} :[0.0, X]. ---(N1).

where X is in the range [0, 1] and ';' denotes disjunction.

An alternative representation of the above is:

non-axisymmetric cracking ===> {S1,S2} :[Sl, 1.0].

where Sl = (1-X), and ===> denotes a causal implication. The latter notation is the one used during the acquisition process as it is more intuitive.

Most of the rules used in negative diagnosis have a single disjunct:

non-axisymmetric cracking ===> {S1} :[Sl, 1.0].

Rules of more than one disjunct arise only in circumstances where (for example) the fault may be triggered by a number of operational events. If the operational events are represented by E1,E2,...,En then rule would be:

fault ===> {E1;E2;..;En} :[a, 1.0].

Unless the addition of an extra disjunct to a rule body significantly reduces the value of X (i.e. makes the rule better) then its introduction is pointless and will generally be distort the evidence.

Suppose the symptoms hypothesised to be useful in negative testing (by the already acquired positive testing knowledge) are:

prog: changes progressive with time,
dwmtemp: changes in a timescale of days, weeks, & months
etc,

then likely rules would be:

non-axisymmetric cracking ===> prog :[0.6, 1].
non-axisymmetric cracking ===> dwmtemp :[0.4, 1].
etc.

Negative testing rules are determined in groups corresponding to description type. For example, *temporal*, symptoms are considered first and a set of rules produced, then the *locational*, etc. For the symptom (or description) type *temporal* acquisition of rules for negative testing proceeds as follows:

start of neg.elicit

Generating rules for refutation of:

non-axisymmetric cracking

select disjuncts to use as the body of a rule:

{<object (e.g. fault)> ===> <disjuncts>:[l,(1-X)]} or

{<object (e.g. fault)> :- NOT <disjuncts>:[0,X]}

1 changes progressive with time

2 changes in a timescale of days, weeks, & months

:1 (a single disjunct rule is selected)

how useful is the rule below as a basis

for eliminating ** non-axisymmetric cracking

non-axisymmetric cracking ===>

changes progressive with time.

0 useless

1 not very useful

2 fairly useful

3 useful

4 very useful

5 superb

N = ?

This gives rise to a rule of the form:

non-axisymmetric cracking ===>

NOT changes progressive with time :[0,X].

where::

N	X
0	0.0
1	0.2
2	0.4
3	0.6
4	0.8
5	1.0

The process repeats from **start of neg.elicit**, using the (so far) unused symptoms, in the above case:

1 (selected previously)
2 **changes in a timescale of days, weeks, & months**

The above process is repeated for each description (*temporal*, *locational*, etc) to enable the elicitation of all useful refutative rules.

Sample output generated by the expert system illustrates the above described process more fully. If further explanation is required then consult the sample output in appendix B.

The disjuncts in the preceding discussion are referred to as symptoms (which in a sense they may be). However this does not necessarily mean that support for them is input directly the user. Support for some of the symptoms will be calculated from 'lower level' symptoms by using other negative and positive testing rules.

Knowledge used to positively and negative test these 'lower level' symptoms is elicited using the same procedure as above.

For example, support for **loss of mass** derives partly from support for **out of balance** and support for **out of balance** derives from support for other quantities. And so on.

Rules (descriptions, measures, and rules of refutation) appear to be an adequate representation for the bulk of the elicited knowledge. However, where support for

one thing derives from the quantity of another thing, knowledge is better represented using FUZZY relations. For example, to determine the support for the statement:

There have been changes in the observed vibrations on several bearings

it is necessary to know the *number* of bearings on which there have been changes.

The expert specifies the level of support assigned to the statement that given numbers of bearings show changes, and thereby creates a FUZZY relation which associates a level of support depending on the number of bearings that are effected. Supports that are not explicitly stated are calculated by interpolation. For example, the supports might given as:

number of bearings	support for: **number = several**
0	[0,0]
2	[0,0]
4	[1,1]
6	[1,1]
14	[0,0]

Then for example, the support associated with the statement that there have been changes on several bearings given that three bearings have been effected is given by:

Support = [0.0, 0.0] + (3-2)/(4-2) * [1.0-0.0, 1.0-0.0]

= [0.5. 0.5].

The above process is slightly complicated by the calculation of how many bearings show a change. For each bearing there will be a support pair specifying the level of support that there has been a change. These supports can be summed as follows to give a measure of the actual number of bearings:

Number of bearings = $\sum^{i=1..n}$ [Sl^i, Su^i]

= [SL, SU]

where n is the number os bearings (usually 14) and
[Sl^i, Su^i] is the support (pair) that there has been a change on the i'th bearing.

[SL, SU] then represents upper and lower bounds on the number of bearings that have shown a change. This is then applied to the relation which yields a support pair which represents the evidence for the statement that there have been changes on several bearings.

For example, if the supports are:

bearing	support for significant change
1	[0.5, 0.6]
2	[0.8, 1.0]
3	[1.0, 1.0]
4	[0.4, 0.4]
5,6,..,14	[0.0, 0.0]

then the sum of the supports is:

[(0.5+0.8+1.0+0.4),(0.6+1.0+1.0+0.4)] = [2.7,3.0],

so the support that there have been changes on several bearings is given by:

SUP= [((0.7/2)*1),((1.0/2)*1)],
= [0.35,0.5].

Relations similar to the above are needed to calculate the supports for other quantities involving the numbers of bearings showing changes (i.e. few, more than one, etc), the speed range over which changes have occurred, the time scales of changes, etc.

5.6 Recap.

This chapter has considered automatic and manual knowledge acquisition, especially from the point of view of the knowledge representation described in chapter 4. It has also considered knowledge acquisition via system modelling.

The manual approach has been the basis of knowledge acquisition for the turbo-generator expert system. Knowledge has been acquired from a mixture of analysis of case histories and expert consultancy. Some examples have been given.

An extensive worked example has been given, which outlines the method of acquisition supported by the current implementation of the expert system. Appendix B comprises sample output from the system which includes a typical knowledge acquisition session.

The following chapter describes the design and implementation of the expert system.

CHAPTER 6

EXPERT SYSTEM DESIGN AND IMPLEMENTATION

This chapter deals with the design and implementation of an expert system for turbo-generator state monitoring.

Section 6.1 contains a preliminary discussion of the basic ideas behind the expert system, from the design point of view.

Sections 6.2. and 6.4 respectively, describe design and implementation of the expert system, and Section 6.3 is a brief user guide.

6.1 A preliminary discussion.

Chapter 4 describes a knowledge representation scheme and associated method of inference, and gives an extensive worked example. An expert system based on this scheme has been implemented.

The following reiterates Chapter 4 by giving a brief outline of the inference method.

The expert system is based around a generate and test paradigm. Hypotheses are first generated (on the basis of data supplied) and then tested against the data. Testing combines 'refutation' and 'confirmation' approaches to give an overall assessment of hypotheses.

Given a hypothetical fault P, refutation is performed first to give an overall support of the form:

P : [0.0, PS] ------(S1)

where PS is the overall possible support for P. This is equivalent to one minus the overall necessary support for (NOT P):

NOT P : [(1.0-PS), 1.0].

If PS is lower than a suitably chosen cut-off value, the hypothesis is assumed to be refuted. If PS is greater than or equal to the cut-off value then positive (confirmation) testing is performed to determine the weight of evidence supporting the hypothesis.

The purpose of using a cut-off value for the necessary support to decide whether to perform positive testing, is simply to make hypothesis testing more efficient. It is reasoned that if opposing evidence is high then supporting evidence will be low therefore obviating the need for positive testing.

Use of the turbo-generator expert system during 'validation testing' has tended to support this view. For each of several realistic sets of symptoms (based on real historical scenarios) the rankings of the likelihood of all possible faults derived from negative testing and positive testing were almost identical.

If positive testing is carried out, the resulting overall support pair is of the form:

P : [NS, 1.0] ------(S2)

where NS is the overall necessary support for P, which represents the overall supporting evidence.

The supports in (S1) and (S2) are combined to give a single overall support for P. One method of combination is to take the intersection of the support pairs (assuming that an intersection exists) to obtain an overall support pair:

P : [NS, PS].

The intersection combination method assumes dependence of proof paths (in this case dependence of supporting and opposing evidence).

If supporting and opposing evidence is assumed to be independent (based on different knowledge and evidence) then the Dempster's rule of combination should be used

instead of the intersection rule. Details of Dempster's rule are given in chapter 3 on Reasoning Under Uncertainty).

The application of evidential reasoning to machine monitoring enables the expert system to accurately assess the likelihood of machine faults given the symptoms. This in turn enables an accurate and consistent remedial strategy.

The weight of evidence for possible faults gives rise to a 'pseudo' cost model in which cost is minimised (or machine utility is maximised: see chapter 1) by choice of remedial actions.

The expert system uses 'symptoms' to diagnose 'faults' and then chooses 'remedial actions' to alleviate the 'faults'.

The expert system is designed to accept details of 'symptoms' by interactive input from a human user. It then performs reasoning to compile a list of faults with associated support pairs. These are sorted into an order based on the weights of related evidence. The fault list is then used in assigning relative 'utilities' to feasible remedial actions.

A 'remedial' action is defined as being any action that will correct a fault or its unwanted side-effects, or tend to reduce uncertainty in the evidence relating to the fault. The latter increases the probability of accurate diagnoses thereby increasing the overall effectiveness of state monitoring.

An important function of an expert system is to give intelligible explanations of reasoning. An expert system should not merely be able to give 'one line answers' to questions, for example:

'The symptoms show that the fault is most likely to be X',

although this is a necessary condition of its usefulness. An expert system should have an extensive and helpful user interface, with an ability to give an adequate degree of explanation of its reasoning. If possible, it should also have the capacity to answer hypothetical ('what if' type) questions.

Both of the above features have been implemented in the turbo-generator expert system. It can give an (if necessary step by step) account of its processes, including an explanation of how it calculates supports.

The system has been designed to receive interactive inputs but could be easily modified to receive inputs from lower level systems (eg. sophisticated data collection devices). In this case, the supports for symptoms would need to be calculated automatically and fed to the expert system.

6.1.1 Interactive vs. On-line:

An interactive expert system is one which interfaces with a human user in order to receive instructions and data, and to output solutions and decisions.

A strictly on-line expert system is one which interfaces directly to other systems possibly without need for operator intervention.

The problem with the on-line approach is that current technology is not sufficiently advanced that expert systems can be left entirely to their own devices. They tend lack the depth of knowledge that is needed to cope with unusual circumstances. Therefore, except in simple knowledge domains, expert systems need to be supervised.

Interactive expert systems.

An interactive expert system is effectively a reasoning aid. It extends the availability of expertise by assisting novice users to perform expert tasks. For example, a General Practitioner has an overview of medicine, understands most medical terminology, and knows how to recognise many clinical signs, but is unlikely to have an in depth specialist knowledge. An expert system can utilise the GP's general skills to gather information on which to base diagnoses, and then to interpret its diagnoses and implement its recommendations.

6.1.2 The structure of an expert system.

Expert systems comprise a kernel or core, and a number of modules associated with the following functions:

1) knowledge acquisition,
2) user interface,
3) data acquisition,
4) help and explanation.

The expert system kernel comprises an inference engine and a knowledge base. The inference engine performs 'mechanical' reasoning using data supplied and knowledge from the knowledge base.

The knowledge acquisition module facilitates the transfer of expertise to the knowledge base from the outside world.

Knowledge acquisition includes addition of new knowledge (as in learning) and maintenance (as in validating and augmenting) of existing knowledge.

The user interface enables easy system operation. It ensures that all output from the system can be readily understood by a user, and that data can be easily input.

The data acquisition module obtains data (to describe a situation) from the outside world by asking questions in such a way that they can be answered easily. Answers are translated to internal form and passed to the system kernel.

The explanation module explains system reasoning. It gives details of:

1) intermediate reasoning steps,
2) the knowledge used in each reasoning step, and
3) the effect of hypothetical data changes on the outcome of reasoning, for example: 'what would happen the support for this symptom were X'.

6.2 Turbo-generator expert system design.

This section describes the design of the turbo-generator expert system in terms of its major components.

6.2.1 The expert system kernel.

The expert system kernel comprises:

1) a knowledge base, and
2) an inference engine.

The knowledge base contains expert knowledge relevant to turbo-generator state monitoring. Knowledge is represented according to the scheme described in chapter 4 (on knowledge representation and inference), and acquired as described in chapter 5 (on knowledge acquisition).

The knowledge base is divided internally according to function. The two main functions of the system are to diagnose faults and recommend remedial actions. The knowledge associated with these is largely independent and is therefore collected and used separately.

The inference engine reasons using knowledge (in the knowledge base) and symptoms supplied from the outside world. Reasoning is performed in a simple, 'mechanical', repeatable way. The method is described in detail in chapter 4 (on knowledge representation and inference).

The inference engine is invoked from the system's menu driven user interface. The overall monitoring strategy may be partly user controlled, but the user has no low level control over inference.

The user interface (described later) allows the user to choose how system tests fault hypotheses and selects remedial actions. It also allows the user to answer hypothetical ('what if' type) questions.

6.2.2 The knowledge acquisition module.

The knowledge acquisition module introduced above, facilitates the transfer of knowledge from expert to the expert system.

Knowledge acquisition includes:

1) constructing a new knowledge base, and
2) maintaining existing knowledge bases.

Setting up a new knowledge base is by far the most complex aspect of knowledge acquisition. Maintaining (eg. adding to) an existing knowledge base is often a simple matter of generalising or copying an existing structure.

The most important function of the knowledge acquisition module in constructing a new knowledge base is to elicit general information about the problem area. This includes:

a) possible faults,
b) symptoms,
c) system operating parameters, conditions etc. and
d) relationships between a) b) and c).

In order to perform a) to d) above, the knowledge acquisition module needs to interface well with the user. Terminology has to be carefully defined at the outset to enable effective, unambiguous communication between expert and expert system.

To enable the addition of new knowledge to an existing knowledge base the acquisition module must initiate a dialogue with the expert, prompting for inputs wherever possible. Received knowledge is translated to internal representation without involving the expert. It may however be necessary to alert the expert to relevant existing knowledge to guard against inconsistency. The system itself performs simple inconsistency checks on its knowledge base.

The knowledge acquisition process is described in detail in chapter 5 (on knowledge acquisition) and an example elicitation session given in appendix B.

Maintenance of existing knowledge is also possible. Existing knowledge is displayed in a clear fashion on the VDU from which is can be changed using a menu driven editor. Editing sessions are monitored to ensure that changes are within the constraints of the representation and that overall consistency is maintained.

6.2.3 The user interface.

The user interface facilitates interaction between the expert system and its user. The user as assumed to have some knowledge of turbo-generator state monitoring.

Experts communicate with the expert system in order to input their expertise, and non-experts communicate with it in order to receive advice.

The user interface is designed to make the system easy to use, so most system facilities can be accessed in more than one way. Novice users can run the system using a nested menu system whilst experienced users can enter mnemonic commands at the system prompt.

6.2.4 The data acquisition module.

The data acquisition module obtains information (about machine behaviour, history, etc.) from a user and converts it to internal form.

To monitor the turbo-generator effectively, the expert system needs to know the weight of evidence for certain symptoms. These are then used to perform fault diagnoses and choose remedial actions.

The data acquisition module asks the user to make qualitative judgements about machine behaviour based on his own observations and assessments of relevant data.

Judgements are input by giving qualitative ascent/descent to assertions made by the system. The order and mode of input is optional.

The exact dialogue depends on the mode of input, but at the lowest level, is essentially the same in each case. The user responds to a question by typing in his

assessment of the evidence. He uses one or two numbers between 0 and 10. These are directly related to the 'support pair'. If only one number is supplied, the support pair is assumed to be of the form:

[X, X]

if two numbers are supplied the support pair is assumed to be of the form:

[X, Y] where X =< Y.

For example, if the only number supplied were N the assumed support pair would be:

[N/10, N/10].

If the two numbers N1 and N2 were supplied the assumed support pair would be:

[N1/10, N2/10].

The user's impressions of the machine are mainly derived from numerical analysis of current and historical data. This is a somewhat 'black art' and even the simple scheme above can appear over precise. Users may feel that there are two or more equally appropriate 'one number' responses. In this case the 'two number' response is appropriate.

There are three modes of data input available to the user, each giving an identical final diagnosis.

In its simplest mode of operation, the system requests data during diagnosis whenever it discovers a lack of relevant input.

The second mode of data input uses the same questioning style as the first but asks questions in a different order. Questions are asked in the order which gives the greatest refutation potential of hypothetical faults which are so far unrefuted.

Suppose for a particular symptom S, and hypothetical faults F1,F2,..,Fn, relevant refutation rules are:

F_1 ===> S :[I_1, 1].
F_2 ===> S :[I_2, 1].
. . . . . .
. . . . . .
F_n ===> S :[I_n, 1].

Given that supports for the hypothetical faults (provided by the currently known symptoms) are:

F_i :[X, Fu_i]. (i=1,2,..,n)

a measure of the symptom's potential for refutation is given by:

'refutation potential' $= \sum^{i=1..n} (Fu_i * I_i)$. --(RP)

Where the symptom does not feature directly in a rule of exactly the above form, but is at the end of a chain of rules of the above form:

F_i ===> S^m :[Sl^m, 1.0].
S^m ===> S^{m-1} :[Sl^{m-1}, 1.0].
S^{m-1} ===> S^{m-2} :[Sl^{m-2}, 1.0].
. . . . . .
S^2 ===> S^1 :[Sl^1, 1.0].
S^1 ===> S :[Sl, 1.0].

the symbol I_i in the expression labelled (RP) above, may be replaced by the expression:

$I_i = Sl^m * Sl^{m-1} * Sl^{m-2} * ... * Sl^2 * Sl^1 * Sl.$

This can be demonstrated from first principals (of 'support logic calculus' - outlined in Chapter 3).

There is an added complication which arises in cases where the body of a rule (of the above form) is disjunctive:

S^j = {X1;X2;X3;..;Xp}

where ';' is an infix operator denoting disjunction.

If the actual rule is:

S^{j+1} ===> {X1;X2;X3;..;XP ;S^j} [XL, 1.0].

then the following rule is assumed:

S^{j+1} ===> S^j :[Sl^{j+1}, 1.0]

where: $Sl^{j+1} = XL/(2^p)$
and : 2^p denotes 2 raised to the power of p.

This rule is based on a maximum entropy assumption.

To the non-expert user, the system's line of questioning may appear complex and therefore difficult to follow. However, this method may be useful for tutorial purposes as it fairly efficient and in some ways resembles the order in which a human expert would request input.

The third mode of data input is the preferred one. The user is asked before the outset of diagnosis, to supply supports for as many symptoms as possible.

As explained in the two preceding chapters, symptoms are divided into 'groups or types:

temporal,
external,
locational,
etc.

Symptoms are requested in one group at a time. For example, the 'temporal' symptoms might be input first, then 'external' symptoms, and so on.

The temporal symptoms, for example, would all be displayed in a 'pseudo' editor from which the current assessment of evidence for each symptom can be clearly seen.

Evidence for a single symptom may be edited by 'selecting' it and typing in revised values. Alternatively the user may select 'automatic' input in which the system addresses each symptom in turn, prompting for new values as it does so.

In addition to the '0 to 10' input scheme there are additional options 'u, n, y'. These correspond to 'uncertain', 'no', and 'yes' which have support pair representation:

u = [0.0, 1.0]
n = [0.0, 0.0]
y = [1.0, 1.0].

This (third) mode of input has the advantage that similar symptoms are queried together and therefore provide context for one another. This helps the user maintain his flow of thought and keep overall consistency between inputs.

In order for the data acquisition module to interface well with the user, enhancements to the expert system will be required. These include an enhanced 'temporal' representation, the need for which will be discussed in the chapter 7.

6.2.5 The explanation module.

The explanation module gives an intelligible description of how it arrives at decisions. The expert system outputs a 'trace' like (following the 'procedural flow' of the reasoning) account of its reasoning.

The trace displays all knowledge as it is used and explains how and why it is using it. Where appropriate, the cumulative effect of reasoning 'to date' is displayed. Since the problem area is inherently uncertain, much of the explanation concentrates on how weights of evidence (or support) are calculated.

The trace is similar in many respects to a PROLOG trace, allowing a user to follow the diagnosis to any required depth and to skip uninteresting parts. Other supported options are: 'break', 'abort', and 'help' which give added flexibility and robustness to the system. For example, help can be called in the middle of an explanation, to find out the meaning of terminology being used. The expert system may also be recursively called using the help utility.

Sample output from the explanation session is given in appendix B.

6.3 A user guide for the expert system.

This section describes the facilities of the expert system, and how they are used.

6.3.1 Background.

The first crude implementation of the turbo-generator expert system used a PC based version of the FRIL language. This was the 'original' FRIL (Fuzzy Relational Inference Language) as conceived by J.F. Baldwin (Baldwin,) which was based on the theory of fuzzy sets.

Briefly, the FRIL language consists of FUZZY 'base' and 'virtual' relations. A base relation comprises a number of clauses and is analagous to a PROLOG predicate comprising only 'fact type' clauses. There is a similar analogy between virtual relations and predicates comprising only 'rule type' clauses.

The 'original' FRIL inference method was entirely different from the method of resolution used by PROLOG. FRIL was supplied with a query comprising a 'target list' and a 'combination of relations'. The query was answered by performing 'join', 'select', and 'project' operations to combine relations in the query and give a base relation that matched the target list. This method was simpler and faster that resolution but the representation was much less flexible. It proved very difficult to implement the necessary procedural parts of the expert system and progress in implementation was therefore slow. The representation of uncertainty also proved to be too inflexible.

The problems with FRIL were not entirely the fault of the language but partly with the implementation of the language that was available. This version of FRIL implemented in FORTH on an IBM PC by a preceding CASE student. There were a number of 'bugs' in the language and the integral programming environment was very limited. This was compounded by deficiencies in the tutorial based 'user guide'.

The most annoying aspect of the PC FRIL system was its handling of virtual relations. These had to be created and edited using an inadequate 'in built' editor that restricted the size of the relation and performed only limited syntax checks. Within this editor, complex virtual relations appeared as an unformatted mass of open and close brackets, variables and atomic identifiers. If the relation was not perfectly correct when submitted to the language compiler (a translator really) an often ambiguous error would be issued by the system and all edits lost.

The final 'nail in the coffin' of the PC based FRIL system was the discovery of a 128 KByte knowledge base size limit. This, combined with the system's difficulty of use and inflexibility, led to a decision to start from fresh using PROLOG.

A much improved version of the expert system was soon implemented using Edinbourgh's 'C PROLOG' on the faculty's VAX 11/750 running the UNIX operating system. The expert system was subsequently transferred to a Systime machine (a VAX 11/750 clone) and then a Gould machine (Somewhat more powerful than the 11/750) each running UNIX. The latest version of the expert system was adapted to Edinbourgh's NIP 1.4 (New Implementation of PROLOG Version 1.4) which gave a significant speed improvement in execution time but greatly increased the time needed to consult (load in) the expert system's program code.

The main part of the expert system, the inference engine, was implemented as an interpreter in PROLOG. This was responsible for performing all reasoning, combination of uncertainty etc. At the time of implementation this seemed the only feasible approach and also made effective explanation a realistic possibility (since PROLOG programs can themselves be interpreted by an interpreter written in PROLOG).

None of the PROLOG interpreters available in the university at the time, included a 'modules' facility. This meant that the whole expert system including knowledge base,

inference engine, and associated modules had to be loaded simultaneously. The loading process (when using NIP) took up to half an hour. For this reason a 'saved state' was created at the end of each session. Although this was very large (about 1.5 MB) it was thought justified because of the time saved.

Newer PROLOG interpreters such as that within the new FRIL language (EQUIPU AIR) have a 'modules' facility, and in any case would load the expert system code in a fraction of the time.

Reimplementation in 'new' FRIL would be a considerable benefits to the expert system. Apart from 'new' FRIL's greater efficiency (improved speed and use of modules), it has some useful additional features. The 'pop-up' windowing facilities for example, are an invaluable aid to implementing help and explanation. Help and explanation could be obtained without losing the current screen full of information and therefore helps the user to remember where he is within a diagnoses.

Further, the expert system could use some FRIL's inbuilt 'support logic reasoning' which is performed at system level (in C rather than FRIL/PROLOG) and would therefore greatly improve efficiency.

6.3.2 Logging on to the turbo-generator expert system.

To start the expert system the user calls PROLOG and then loads a 'boot-up' file which instructs PROLOG to load all files needed by the expert system.

If a saved state exists the user may simply reactivate the expert system using a one line UNIX command.

In either case when the system is ready it displays the prompt:

ok :

This indicates that the expert system is waiting for a command.

Each command consists of a string of characters followed by a <return>. The following is a list of valid commands that may be issued at the 'ok :' and other

prompts. The 'help' command could, for example, be given at any prompt during knowledge acquisition or whilst performing a diagnosis.

In the following '< X >' denotes an instant of X. So for example, 'fault' and 'description' identifiers are denoted as <fault> and <description>. 'Non-axisymmetric rotor crack' is a possible value for <fault>.

To refer to a particular fault, symptom or description, one of two possible identifiers may be used. One is an abbreviated form of the other, eg:

'nacrack' or 'non-axisymmetric rotor crack'.

Notice that the first of these is an atomic descriptor containing no spaces. The second is a short natural language equivalent.

The actual identifiers known to the system are defined by the domain expert, but may be changed by the user if desired.

6.3.3 Expert system commands.

help calls up the 'help menu' from which help on terminology, faults, symptoms, etc. may be obtained.

help <identifier> causes a help on <identifier> to be displayed. <identifier> can be any fault, symptom, or description.

edit help takes the user to the 'edit-help menu', from which he may edit help information relating to faults, symptoms, etc.

edit help <identifier> allows the user to edit the help information on <identifier>.

check tells the system to check its knowledge base and display anything that is either syntactically incorrect or inconsistent with the rest of the knowledge base.

user calls up the utilities menu. The following options are included:

1) abort execution of PROLOG
2) break execution of PROLOG
3) look at PROLOG statistics
4) start a recursive UNIX shell
5) access UNIX commands menu
6) perform a visual edit of a file
7) load a file.

These are only made available to the user for the sake of completeness. Their use is mainly for expert system maintenance (eg. modification to the inference engine, editor, user interface, etc.).

vi <file name> allows the user to perform a visual edit of the file <file name>. The expert system has a built in directory system so that system files can be addressed by their name alone irrespective of their location within the UNIX directory structure. For example, the file new.es/syst.preds/check can be edited simply by issuing the command vi check. If the file <file name> is not a system file then it is assumed to be in the default directory. On completion of a visual edit the user may optionally reconsult the file.

more <file name> is very similar to 'vi <file name>' but instead of calling an editor it scrolls the file on the VDU. No reconsult option is available.

load <file name> loads the file: <file name>. As with the above 'vi and more' commands, load operates via the internal directory system.

trace on and trace off switch the trace on and off. If the command trace on is issued then all subsequent diagnoses will be explained, until the command trace off is issued. The converse also applies.

The trace on and trace off commands will not be accepted during explanation of a diagnosis. If explanation is unintentionally started, output can be minimised using the trace 'skip' option, or stopped using the 'b' (break) option.

shell calls up the expert system's main menu. This is at the 'apex' of the menu system. All the expert system's facilities can be accessed from the shell menu by moving down through the 'lower' menus. Access is achieved by choosing appropriate options from each menu displayed. This is a common feature of many popular commercial software packages such as 'Lotus 123'.

The menu system will be explained more fully later in this section.

input or ? calls up the 'edit symptoms' menu which controls data acquisition. Menu options enable:

1) input of symptoms (or rather their supports),
2) updating and editing of symptoms,
3) loading and saving complete sets of symptoms,.

input <identifier> or ?<identifier> has a dual function. Its main function is to by-pass the 'edit symptoms' and allow <identifier> to be edited directly.

If <identifier> is a symptom it is edited directly. If it is a description then a menu of associated symptoms is presented for editing. For example, if <identifier> = 'temporal' then the system presents a menu of all possible temporal symptoms.

The second function of the command applies when <identifier> is the name of a fault or high level symptom (one whose support is calculated rather than input). The command causes <identifier> to be tested as a hypothesis. The results of negative, positive, and full testing are displayed. For example:

```
ok : ?loss of mass
neg evidence        :[0.000,1.000]
pos evidence        :[0.260,1.000]
full evidence       :[0.260,1.000]
ok :
```

ntest, ptest and ntest <identifier> perform (respectively) negative, positive, and full tests on <identifier>. See chapter 4 for a description of the different types of testing.

-h +h and *h perform (respectively) negative, positive, and full tests on current hypotheses.

remedy calls up the 'remedial' menu which controls selection of remedial actions. The options available are described later in this chapter.

edit calls up the 'edit knowledge' menu. This allows the user to edit diagnostic knowledge about faults and high level symptoms.

The user is presented with a menu of currently known faults and symptoms whose associated knowledge can be edited by making appropriate selections. The menu can itself be edited (new faults may for example be added and then edited).

The editors response to a selection will depend on the nature of the item selected and the current state of its associated knowledge (eg. whether the knowledge is rule based or relational).

The knowledge acquisition module and editor are described more fully later in this section. Briefly, if relevant knowledge already exists it is checked to ensure that it is intact and complete. If it is incomplete the knowledge acquisition module is invoked to fill in the gaps. If it is complete the knowledge editor is invoked.

edit <identifier> where <identifier> is the name of a fault or symptom, is essentially the same as edit above, but by-passes the first menu.

remove calls up a menu from which the user may select items to be deleted. For example all reference to a certain fault or symptom could be deleted from the knowledge base.

remove <identifier> is similar to remove except that it by-passes the initial menu selection stage and deletes all knowledge relating (specifically) to <identifier> from the knowledge base.

re-solve forces the system to delete any stored supports calculated during past diagnoses. This is done automatically whenever knowledge or symptoms are edited. During a diagnosis, calculated supports are asserted in the data base to avoid duplicating calculations. This can greatly improve the speed of inference.

abort aborts PROLOG execution (and therefore execution of the expert system).

break temporarily suspends PROLOG execution (and therefore execution of the expert system).

x <unix command> chains <command> out to the operating system.

quit terminates the current session and performs a backup of the expert system's knowledge base. This is to insure against accidental deletion of knowledge during subsequent sessions.

6.3.4 The menu system.

The menu system is an integral part of the expert system's user interface. At each menu the user is presented with a list of options. When an option is selected, a sequence of goals is executed by the expert system and then the menu is redisplayed. The menu is exited by pressing <return>. Any valid command may be entered at a menu prompt.

The main part of the user interface (devised for novice users) comprises a set of nested menus. These allow all the expert system's facilities to be accessed by moving about in the menu system and selecting the appropriate menu options.

The diagram below shows a typical 'state of play' in the menu system:

expert edit edit symptoms

page 1

1 input/edit symptoms
2 load a file of symptoms
:

At the top of the screen is a list of menu names which should be read from left to right. These give the current position within the menu system. The current menu 'edit symptoms' is shown last and the first menu 'expert' is shown first, etc.

The above menu allows three options:

1) select option 1
2) select option 2
3) return to the menu above (namely the 'edit menu').

'page 1' in the diagram indicates that page one of the available options is currently displayed. This feature is only important when there are more than (arbitrarily) ten options in which case the menu is divided up into pages.

Each menu is like a mini screen editor. Each menu item is assigned a page which can be displayed, searched, and edited (elements of a page may be selected thus allowing the editor to act as a menu):

f and b scroll forwards and backwards one page.

fn and bn (where n is an integer) scroll forwards and backwards n pages.

/<identifier> searches for <identifier> and displays the next page that contains it. n searches for the next occurrence of the last search string.

en allows line n of the current page to be edited.

an and in 'append a new line after' and 'insert a new line before' line number n of the current page.

dn deletes line n of the current page.

n 'selects' line n of the current page (selecting an item more than once has no additional effect).

<expert system command> causes the expert system to execute the corresponding expert system command. Afterwards the current page is redisplayed. Eg:

: trace on <return>

6.4 Implementation in PROLOG of the expert system.

This section describes the methodology and implementation (in PROLOG) of the expert system. The section is divided into subsections corresponding to the expert system components: kernel and associated modules.

6.4.1 Implementation overview.

The expert system is implemented in 'standard' PROLOG, but this fact is transparent to the user.

The aim has been to produce an efficient system which is fast and easy to use.

The knowledge representation described in Chapter 4, allows knowledge to be represented simply and compactly, and inference can be performed much more quickly than might be expected. For example, the execution time for a full diagnosis (on a VAX 11/750 - giving only 0.7 VAX mips) is about 30 seconds. For an interpreter (written in PROLOG - itself an interpreted language) this is very good, and should be acceptable to most users.

The user interface, data and knowledge acquisition modules, help and explanation also perform with similar speed. Each is written so that 'disasters' such as computer

failure will not result in much loss of expert of operator effort. For example, only incremental changes to the knowledge base can be lost, as the expert only edits a 'copy' of the relevant part of the knowledge base. When editing is complete the copy becomes the 'current' version, and the superseded version is kept as a backup.

The PROLOG code has been written for clarity as well as speed. There are certainly many places where speed could have been improved at the sacrifice of clarity, but this was avoided as it would hinder modification and maintenance.

The PROLOG 'assert' and 'retract' predicates are avoided wherever possible, except in the knowledge and data acquisition modules in which they have an important and intentional use. Use of 'backtracking' and 'list processing' are usually more tidy although (depending on the PROLOG interpreter) sometimes less efficient.

6.4.2 Implementation of the kernel.

The kernel is implemented as an interpreter in PROLOG. For example, when asked to determine the evidence associated with a possible fault, it searches the knowledge base for relevant knowledge. It should find a number of rules potentially giving opposing evidence, and a description of the fault from which to derive supporting evidence.

The mechanisms by which the system derives weights of evidence are fully described in Chapter 4 (on Knowledge Representation and Inference) and are not be repeated here in detail. In summary:

Rules giving opposing evidence (for x) are of the form:

x :- NOT (y1;y2;..;yn) : [0.0, P]

where yi (i=1..n) are symptomatic of x and where ';' is an infix operator denoting disjunction. P is the possible support for x given the body of the rule.

To evaluate this rule, the interpreter tries in turn (from left to right) to derive supports for each yi (i=1..n). It does this using the same mechanism by which it

calculate the support for x. In this respect the interpreter works similarly to PROLOG itself, although it uses no backtracking or instantiation of variables.

When the interpreter can not find a rule associated with a disjunct it is assumed to be a fuzzy set. If no fuzzy set exists then it is assumed to be a symptom that is input by the user.

If there is no evidence relating to the symptom, the interpreter looks to see whether it has been told that it has all available evidence. If so, it assumes the support to be [0.0, 1.0] (ie. uncertain) otherwise it requests input from the user.

Derivation of supporting evidence is similarly performed. The interpreter backwards chains through the relevant rules to the symptom level. However, the approach differs somewhat because 'positive testing' rules are not stored directly. The interpreter has to construct the rules on the basis of fault descriptions in such a way as to optimise the use of available positive evidence (this process is fully explained in chapter 4).

Opposing and supporting evidence is combined at the 'top level' using Dempster's rule to give an overall support pair.

6.4.3 Implementation of user interface.

The user interface is implemented as an 'extended menu' system which has essentially two modes of operation, firstly as a menu system and secondly as an editor.

When the system is operating as a menu, the user is presented with a list of options from which selections can be made. This enable the user to issue instructions to the expert system.

For example, within the knowledge acquisition process, the menu system enables the user to select a number of items from a list. An identifier for each selected item is retained in a list and then passed back to the calling level of the menu.

For example, if the system asks:

'which of the following descriptions apply to the (to be added) fault ?',

the (expert) user selects the relevant descriptions from the list and exits from the menu. The list of selected 'description identifiers' is passed back to the knowledge acquisition module which then obtains details for each selected description.

The extended menu system also contains a 'command line trap' that intercepts certain key commands and performs them directly.

The 'command line trap' is implemented via a predicate that reads input from the keyboard. If the input matches one of the 'key templates' then the command is executed directly and a 'dummy command' passed back to the 'extended menu'. If there is no match the command is assumed to be a valid 'extended menu' command, and is ignored by the command line trap.

One important feature of the trap is that it allows certain commands to be issued from almost anywhere in the system. For example, the diagnostic mechanism could be invoked from within the knowledge acquisition module to investigate the effects of hypothetical changes to the supports on a certain data.

When the system is operating in editor mode, the user can edit tabulated information presented to him by issuing appropriate editing commands (which were described earlier).

The 'extended menu' system is implemented as a clause editor. Clauses which match given specifications are 'picked up' from the PROLOG database and entered in a 'clause list'.

If the 'extended menu' is being used as an editor then 'picking up' the clauses involves retracting them from the database on entry and reasserting them on exit (in modified form). If the 'extended menu' is being used as a menu then clauses are not changed and there is no need to retract and reassert them.

On entry to the menu, the 'clause list' is divided into pages of predefined length. Each page is numbered, as are the clauses on each page. One page is displayed at a time, and the user moves between pages by issuing a direct command or by searching for an item. Each command issued is referred (initially at least) to the currently displayed page.

Each clause is displayed as specified in the knowledge base. In general, only a subset of the arguments of a clause are shown. Each argument may be displayed 'as is' or in an 'extended form'. For example, na_crack denotes 'non-axisymmetric crack' and either may be displayed.

To help the user keep track of 'where' he is within the expert system and which function is currently being performed, a message is displayed at the top of each screen. The message comprises a heading (with brief instructions to the user) and a list of the calling hierarchy. This shows the user where control will be returned to on exit.

For example, if the current menu 'add new knowledge' were called from the menu 'modify knowledge base' which was itself called from the main menu 'main' the following list would be displayed at the top of the screen:

main modify knowledge base add new knowledge

The 'extended menu' system and 'command line trap' are both implemented so that new commands easily and safely added without need to greatly modify existing PROLOG code. For example, to add new commands to the editor a programmer need only add new code by copying the existing, clear structures.

6.4.4 Implementation of data acquisition.

Data acquisition can be performed in one of three ways:

1) all available data is input before outset of diagnosis,

2) some data is input at the outset, and the system then requests input when it finds a 'gap' in the data it requires,

3) the system performs a diagnosis in a way that minimises data input, requesting this data as diagnosis proceeds (the disadvantage of this method is that requires greater processing and is therefore slower).

6.4.5 Implementation of knowledge acquisition.

Knowledge acquisition is performed using question and answer sessions. The user selects the nature of the knowledge to be added and the system asks questions to determine the details. Each question is posed so that it can be answered by selecting one (or more) of a number of options shown to the user. This is achieved using the 'extended menu' system described above.

For example, qualitative opinions (about the importance of a particular description of a fault - say) are specified by the user by selecting a menu option which best reflects his opinion.

6.4.6 Implementation of explanation.

Explanation is implemented as a form of trace facility which may be switched on or off. It works in a similar way to the PROLOG trace, and has similar features and permissible commands.

Essentially, the trace (when switched on) follows the internal workings of the expert system kernel and outputs a commentary to the screen to inform the user what functions are being performed, with what data, and with what results.

The 'level' of commentary can be controlled using the 'skip' command which is similar to that used in the PROLOG. For example, the system evaluates a rule by evaluating each conjunct in its body and combining the results. If the user is only interested in how one of the conjuncts is evaluated, and in the overall result, he may skip the trace of the uninteresting conjuncts and therefore only look at one in detail. The final stage of combining the evidence is performed automatically and the result displayed.

Explanation is in almost natural language form and is therefore easily intelligible. Sample output is given in Appendix B.

As explained in an earlier section, the kernel is implemented as an interpreter written in PROLOG. Explanation is enabled by interpreting the 'kernel interpreter' and displaying suitable commentaries. When the explanation is switched off, the 'kernel interpreter' runs directly from PROLOG, but when explanation is switched on, a 'meta' interpreter is invoked.

The interpreter runs the 'kernel interpreter', and as it does so it gets instructions from the user (on how to conduct the explanation), and executes 'other PROLOG code' (in parallel) to output messages which form an explanation.

The explanation interpreter is written so that the 'other PROLOG code' is executed wherever it exists (unless instructed to skip it by the user). If it does not exist the interpreter skips explanation and passes control for that process (performed by a single predicate) to PROLOG itself. This is to enable the explanation facilities to be easily maintained (enhanced etc).

6.4.7 Implementation of help.

Help is implemented using the menu system. When help is invoked, the help menu is called up from which the user may obtain help. Help is subdivided in to subject types. For example, there is help on 'terminology', 'use of the menus', 'use of the command line', etc.

Provision has been made to enable context sensitive help. This is done through the predicate that reads from the keyboard (and implements the 'command line trap'). For example, if help were requested when the system was reading the name of a fault it could output a message detailing names already in use and permissible names etc.

CHAPTER 7

CONCLUSIONS AND FURTHER WORK

7.1 Conclusions.

This thesis has described work done in producing an expert system for the state monitoring of turbo-generators.

Early work concentrated on gaining an understanding of the problem domain, through discussions with experts at the CEGB, and analysis of documented case histories.

This work revealed that a central problem in the task domain would be to find an efficient way of handling uncertainty.

The initial plan was to implement the expert system using a language called FRIL. This is quite different from the current version of FRIL which is based on a theory of support logic. The 'original' FRIL language was based on fuzzy set theory and knowledge was represented by means of fuzzy base and virtual relations.

The 'original' FRIL system was found to be inadequate for the task, although this was due in part to the poor implementation of the language and its limited environment.

The fuzzy set theory representation was also inadequate for the task by virtue of being too rigid and not sufficiently expressive, especially from the point of view of modelling uncertainty.

This motivated the invention of a new model of uncertainty which is similar in many respects to support logic, but represents knowledge at a slightly different level.

Knowledge is represented so that diagnostic rules with many antecedents, can be constructed from descriptions of faults and ways of measuring those descriptions using symptoms. This ensures that evidence is used effectively, and overcomes the problems of having rigid rules. For example, the following support logic rule assigns

support to p, depending on the supports for the conjuncts q1, q2, q3, and q4, and on the conditional support pair [Cl, Cu]:

p :- q1,q2,q3,q4 : [Cl, Cu].

However, if any one of q1,q2,q3,q4 has a [0.0, 1.0] support, no matter what the support for the remaining 3, then the rule is effectively redundant. This is overcome in support logic by giving alternative rules such as:

p :- q2,q3,q4 : [Cl1, Cu1].

p :- q1,q3,q4 : [Cl2, Cu2].

p :- q1,q2,q4 : [Cl3, Cu3].

p :- q1,q2,q3 : [Cl4, Cu4].

etc.

All such rules are evaluated, and the intersection of the resulting support pairs is taken as the support for p.

This technique is only effective when there are small numbers of conjuncts in a rule. If there were 8, for example, then it would impractical to try and list the necessary alternative rules (up to 2^8 rules would be required).

The new model of uncertainty forms the basis of an expert system written in PROLOG. Reasoning is performed by means of a backward chaining rule interpreter which assigns support to hypothetical faults on the basis of supported symptoms supplied by a user. Both supporting and opposing evidence is calculated and then combined to give an overall assessment of the evidence.

On the basis of its assessment of fault hypotheses the expert system can then suggest to the user, an optimal strategy for acquiring additional data. This takes into account the potential costs associated with the hypothetical faults, and the effects of reducing existing evidential uncertainties in data on the assessment of hypotheses.

The expert system is fully menu driven from a computer terminal and includes facilities for knowledge acquisition and maintenance, explanation of reasoning, and getting help on use of the system.

Testing of the system has only been on an informal basis. However, in all cases where the system has been given 'real' problems by experts from the CEGB, it has given correct or at least plausible diagnoses.

7.2 Further work.

Further work in needed to extend and evaluate the expert system in the following respects.

7.2.1 Reimplementation in a more flexible environment.

The PROLOG based system would benefit from being reimplemented in a more up to date programming environment such as that provided by the new FRIL language. Some of the inference could then be performed directly in FRIL rather than through an interpreter, as this is rather clumsy and inefficient. Furthermore, FRIL incorporates facilities for screen handling which are invaluable for building a user interface. For example, this could enable help and explanation to be displayed to the user without loosing existing screen information.

7.2.2 Validation and testing.

The current system needs to be tested to assess its performance and to validate its knowledge base by giving it realistic problems. This would need to be done in cooperation with the CEGB, and should reveal any weaknesses in the knowledge representation and method of inference, and uncover any bugs and gaps in the knowledge base.

7.2.3 Temporal extension to the knowledge representation.

The existing knowledge representation would benefit from being extended to include the timing duration of machine behaviour and operational events. This would enable the expert system to perform reasoning at a more fundamental level, and make it less dependant on its operator.

For example, in order to determine whether a change in machine vibrations could have been due to an operational event, the system would currently have to ask a question such as "was there an operational event X prior to the change in machine vibrations". This is slightly unsatisfactory as it relies on the operator making expert judgements as to whether the change in machine vibrations occurred within a sufficiently short time of an operational event for it to have been their cause.

However, if all events and machine behaviour were given a time referent comprising a start and end time, or a start time and duration, then the system could make its own temporal judgements and the importance of the operator would be reduced and possibly obviated.

REFERENCES

Allen J.F, (1981), "An internal-Based Representation of Temporal Knowledge", Proc 7th IJCAI, pp 221-226.

Allen J.F & Kooman, (1983), "Planning Using a Temporal World Model", Proc 8th IJCAI, pp 741-747.

Allen P, (1983), "Internal CEGB report Ref: SWR/SSD/D/1551/S/83", South Western Regional Headquarters, CEGB.

Bannister R.L & Osborne R.L & Jennings S.J, (Jan 1979), "Modern Diagnostic Techniques Improve Steam-Turbine Reliability", Westinghouse Electric Corp. Power, Turbines and Diesels / Instrumentation and Control / Energy Management, pp 46-50.

Baldwin J.F, (1979), "A new Approach to Approximate Reasoning using a Fuzzy Logic", Fuzzy Sets & Systems, 2, pp 309-325.

Baldwin J.F, (1979), "Fuzzy Logic and Fuzzy Reasoning", Int Jnl Man-Machine Studies, 11, pp 465-480.

Baldwin J.F, (1982), "An Automated Fuzzy Reasoning Algorithm", in Yager R.R (ed), Recent Advances in Fuzzy Sets and Possibility Theory, Pergamon Press, NY.

Baldwin J.F, (1983), "A knowledge Engineering Fuzzy Inference Language - FRIL", in Proc of AUWE Knowledge Eng Conf.

Baldwin J.F, (1986), "Support Logic Programming", in Jones A.I et al, Eds, Fuzzy Sets Theory and Applications, (Reidel, Dordrecht-Boston).

Baldwin J.F et al, (1987), "Evidential Support Logic Programming", Fuzzy Sets and Systems, 24, pp 1-26.

Baldwin J.F et al, (1987), "FRIL Manual", FRIL Systems Ltd, St Anne's Rd, Bristol BS4 4A, UK.

Baldwin J.F & Pilsworth B.W, (1980), "Axiomatic Approach to Implication for Approximate Reasoning with Fuzzy Logic", Fuzzy Sets and Systems, 3, 193-219.

Baldwin J.F & Pilsworth B.W, (1982), "An Inferential Fuzzy Logic Knowledge base", Proc Workshop on Logic Programming for Intelligent Systems, RMS Queen Mary, Long Beach, Calif USA.

Baldwin J.F & Zhou S.Q, (1983), "An Implementation of FRIL", Univ of Bristol, Dept of Eng maths, Internal report Ref: EM/FS 46.

Bannister D & Fransella F, (1971), Inquiring Man: The Theory of Personal Constructs, Penguin Modern Psychology.

van Benthem J, (1983), The Logic of Time, Reidel.

Bochvar D, (1939), "On Three-Valued Logical Calculus and its Application to the Analysis of Contradictions", Matematiceskij Sbornik, 4, pp 353-369.

Chandrasekaran B & Tanner C, (1986), "Uncertainty Handling in Expert Systems: Uniform vs. Task-Specific Formalisms", Uncertainty in Artificial Intelligence, Machine Intelligence and Pattern Recognition, 4, North-Holland, pp 35-46.

Clark K.L & McCabe F.G, (1984), Micro-PROLOG: Programming in Logic, Prentice Hall.

Clark K.L & Tarnlund S.A (eds), (1981), Logic Programming, Acad Press.

Clocksin W.F & Mellish C.S, (1981), Programming in PROLOG, Springer Verlag.

Cox R.T, 1946, Probability, Frequency and Reasonable Expectation, American Journal of Physics, 14, pp 1-13.

Cox R.T, 1979, "OF Inference and Inquiry - An Essay in Inductive Logic", In The Maximum Entropy Formalism, Ed. Levine and Tribus, M.I.T Press.

Davis M, (1980), "The Mathematics of Non-monotonic Reasoning", Artificial Intelligence, 13, pp 73-80.

Dempster A.P, (1967), Upper and Lower Probabilities Induced by a Multivalued Mapping, Annals of Mathematical Statistics.

Dempster A.P, (1968), "A generalisation of Bayesian Inference", J. of Royal Statistical Society, series B, 30.

Doyle J, (1979), "A Truth Maintenance System", Artificial Intelligence 12, pp 231-272.

Dubois D & Prade H, (1985), "Fuzzy Cardinality and the Modelling of Imprecise Quantification", Fuzzy Sets and Systems, 16, pp 199-230.

Filman R.E, (1979), "The Interaction of Observation and Inference in a Formal Representation System", Proc 6th IJCAL, pp 269-274.

Garvey T.D & Lawrance J.D & Fischler M.A, (1981), "An Inference Technique for Integrating Knowledge from Disparate Sources, Proc IJCAI '81, pp 319-325.

Haack S, (1974), Deviant Logic, Cambridge Univ Press.

Hayes P.J, (1977), "In Defence of Logic", Proc 5th IJCAL, pp 559-565.

Hayes P.J, (1977), "On Semantic Nets, Frames and Associations", Proc 5th IJCAL, pp 99-107.

Jakob F & Suslenschi P & Vernet D, (1986), "EXTASE: an Expert System for Alarm Processing in Process Control", Proc 7th ECAI, Vol 2, pp 103-108.

Kelly G.A, (1955), "The Psychology of Personal Constructs", Vols 1 & 2, Norton.

Kelly G.A, (1961), "The Abstraction of Human Processes", Proceedings of the 14th International Congress of Applied Psychology, Munksgaard, Copenhagen.

Konolige K, (1982), "A First Order Formalisation of Knowledge and Action for a Multi-Agent Planning System", in Hayes J.E & Michie D & Pao Y.H, (eds), Machine Intelligence 10, pp 41-72, Ellis Horwood.

Kubiak J.A & Rothhirsch A.L & Aguire J.R, (1984), "An Algorithm of Fault Diagnosis for Turbine Operations", Instituto de Investigaciones Electricas. Cuernavaca, Mexico.

Kurihara N & Nishikawa M & Nagahashi Y, (1983), "A Microcomputer Based Vibration Diagnostic System for Steam Turbines and Turbo-Generators in Power Plants", Hitachi Research Lab., Hitachi Works, Hitachi Ltd, Ibaraki, Japan, Proc IEEE Ref: 83 SM 487-6.

Lukasiewicz J, (1920), "On Three-Valued Logic", reprinted in McCall (ed), Polish Logic (1920-1939), Oxford Univ Press (1967).

McArthur J, (1979), Tense Logic, Reidel.

McCarthy J, (1977), "Epistemological Problems of Artificial Intelligence", Proc 5th IJCAL, pp 1038-1044.

McDermott D, (1982), "A Temporal Logic for Reasoning about Plans and Actions", Cognitive Sciences, 6, pp 101-155.

McDermott D, (1982), "Mon-monotonic Logic II", J Assoc Computing Machinery, 29, pp 35-57.

McDermott D & Doyle J, (1980), "Mon-monotonic Logic I", Artificial Intelligence, 13, pp 41-72.

Myers J.A, (1982), "Internal CEGB report Ref: SSD/SW/82/M18484", South Western Regional Headquarters, CEGB.

Minsky M, (1974), "A Framework for Representing Knowledge", MIT, AI Memo 306.

Minsky M, (1982), "Why People Think Computers Can't", AI Magasine 3, pp 1-15.

Moore, R.C, (1984), "A Formal Theory of Knowledge and Action", in Hobbs J.R & Moore R.C, Formal Theories of the Commonsense World, Ablex Press.

Moszkowski B.C, (1985), "A Temporal Logic for Multilevel reasoning about hardware", Computer, 18, 2, pp 10-19.

Moszkowski B.C, (1986), Executing Temporal Logic Programs, Cambridge Univ Press.

Myers J.A, (1983), "Internal CEGB report Ref: SWR/SSD/D/1582/S/83", South Western Regional Headquarters, CEGB.

Myers J.A, (1984), "Internal CEGB report Ref: SWR/SSD/D/1660/S/84", South Western Regional Headquarters, CEGB.

Myers J.A & Thomas D.L, (1982), "Internal CEGB report Ref: SWR/SSD/SW/81/N211", South Western Regional Headquarters, CEGB.

Nichols T, (1984), "Internal CEGB report Ref: SWR/SSD/D/1604/S/84", South Western Regional Headquarters, CEGB.

Nichols T, (1984), "Internal CEGB report Ref: SWR/SSD/D/1665/S/84", South Western Regional Headquarters, CEGB.

Nichols T, (1984), "Internal CEGB report Ref: SWR/SSD/D/1685/S/84", South Western Regional Headquarters, CEGB.

Pegrum L.J, (1984), "FRIL on The IBM Personal Computer", Univ of Bristol, Dept of Eng Maths, FRIL User Guide.

Prade H, (1985), "A Combinatorial Approach to Approximate and Plausible Reasoning with Applications to Expert Systems", IEEE Trans on PAMI, Vol 7, No. 3, pp 260-283.

Rescher N, (1969), Many-Valued Logic, McGraw Hill.

Rescher J & Urquhart A, (1971), Temporal Logic, Springer Verlag.

Roth A & Brooker M, Eds., (1988), "Diagnostic Expert System Generates Powerful Saving", Expert System User, 4, 4, pp 8-9.

Shafer G, (1976), "A Mathematical of Evidence", Princeton Univ Press.

Shank R.C (ed), (1975), Conceptual Information Processing, North-Holland.

Shatoff J, (1976), "Using Vibration Analysis to Determine the Dynamic Health of Turbine/Generators", Westinghouse Electric Corporation, Power, 120, 5, pp 23-28.

Sowa J.F, (1984), Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley.

Turner R, (1984), Logics for Artificial Intelligence, Ellis Horwood.

Weyhrauch R.W, (1878), "Prolegomena to a Theory of Mechanised Formal Reasoning", Memo 315, AI Lab, Stanford Univ.

Woods W.A, (1975), "What's in a Link: Foundations for Semantic Networks", in Bobrow D.G & Collins A (eds), Representation and Understanding, Acad Press.

Zadeh L.A, (1965), "Fuzzy Sets", Information & Control, 8, 338-353.

Zadeh L.A, (1978), "Fuzzy Sets as a Basis for a Theory of Possibility", Fuzzy Sets and Systems, 1, pp 3-28.

Zadeh L.A, (1983), "Commonsense Knowledge Representation Based on Fuzzy Logic", Computer, 16, pp 61-65.

Zadeh L.A, (1986), "Is Probability Theory Sufficient for Dealing with Uncertainty in AI: A Negative View", Uncertainty in Artificial Intelligence, Machine Intelligence and Pattern Recognition, 4, North-Holland, pp 103-116.

Harmon P, Maus R & Morrissey W, (1988), "Expert Systems : Tools & Applications", Wiley

APPENDIX A

THE KNOWLEDGE BASE

This appendix contains all knowledge relating to turbo-generator state monitoring that has been acquired and entered into the expert system.

The 23 different predicates listed below are in exactly the form in which they are loaded into PROLOG and used by te expert system.

The interpretation of the predicates below, which are greatest importance, are described in chapter 4 on knowledge representation and inference.

Just before the listing of each predicate, there is a short section which describes it briefly.

The 'object' predicate:

The 'object' predicate represents knowledge about objects, including a textual description of the object, and whether the object is a fault or a symptom.

Clauses of the 'object' predicate are of the form:

object(LEVEL,TEXT,ATOM,TYPE)

where:

1) LEVEL represents the level of the object, eg. high or low. High indicates that evidence for this object derives from calculations based on evidence relating to other object.

2) TEXT is a textual description of the object. Eg. 'loss of mass' or 'thermal bend'.

3) ATOM is an atomic identifier (a continuous string of characters) which is a shortened version of TEXT.

4) TYPE is the type of the object. Eg. the object 'step change' is an object of type 'temporal (symptom)', and 'loss of mass' is an object of type 'fault'.

The 'object' predicate is as follows:

```
object(l,'slower change has small str of steps', smstrsteps,temporal).
object(l,'fast changes followed by a step', fthenstep,temporal).
object(l,'changes permanent since some datum', perm,temporal).
object(l,'very slow followed slow changes', vsthens,temporal).
object(l,'changes progressive with time',prog,temporal).
object(l,'changes cyclical with time',cyc,temporal).
object(l,'time taken',time,temporal).
object(l,'excursion of gland steam temperature', exgstemp,nonvib).
object(l,'magnetic field disruption',mfd,nonvib).
object(l,'jacking oil pressure',jackop,nonvib).
object(l,'pedestal tilt',pedtilt,nonvib).
object(l,'lost motion',lostm,nonvib).
object(l,'opposing changes in ped vibs and eccentricity', dpviecc,eoh).
object(l,'ft event or op-conds for sticking windings', fteostw,eoh).
object(l,'changes in behaviour preceeded by a run-down', rd,eoh).
object(l,'changes in behaviour preceeded by a run-up', ru,eoh).
object(l,'run-up or early loading restricted', ruelrest,eoh).
object(l,'i squared and step with hysterisis', rc2swh,eoh).
object(l,'changes reversible with run-down',revrd,eoh).
object(l,'permanent bend inducing event',pbindev,eoh).
object(l,'i squared characteristic',rc2,eoh).
object(l,'too fast a run-up',toofru,eoh).
object(l,'recip changes at opp ends of a brg boat', brgbrc,location).
object(l,'number of bearings',brgnof,location).
object(l,'changes on the high pressure turbine', hp,location).
object(l,'changes on the intermediate pressure turbine', ip,location).
object(l,'changes on the low pressure 1 turbine', lp1,location).
object(l,'changes on the low pressure 2 turbine',lp2,location).
object(l,'changes on the low pressure 3 turbine',lp3,location).
object(l,'changes on the generator rotor',gen,location).
object(l,'changes on the exciter',exc,location).
object(l,'changes at crit speeds and sub-multiples', cssmcs,speed).
object(l,'changes in critical speeds',ccs,speed).
object(l,'percentage of speed range',srper,speed).
object(l,'eccentricity at low rpms',rpmslecc,speed).
object(l,'1 and 2 per rev sr correspondance', srcor12,speed).
object(l,'erratic behaviour of once per rev response', oprerat,freq).
object(l,'changes in sub-synchronous response', subsynch,freq).
object(l,'change in response at twice per revolution', tpr,freq).
object(l,'change in response at once per revolution', opr,freq).
object(h,'alignment change',align,fault).
object(h,'example fault',exf,fault).
object(h,looseness,loose,fault).
object(h,'localised looseness',locloose,fault).
object(h,'non axisymmetric cracking',trotcrack,fault).
object(h,'shorted turn',shturn,fault).
object(h,'sticking windings',stwinds,fault).
object(h,'loss of mass',lm,fault).
object(h,'movement of mass',mm,fault).
object(h,rub,rub,fault).
object(h,'permanent bend',permbend,fault).
```

```
object(h,'asymmetric cooling',ascooling,fault).
object(h,'oil in the bore',oilinbore,fault).
object(h,'loss of axial clearance',lossaxclr,fault).
object(h,'out of ballance',oob,fault).
object(h,'changes in tscale of days weeks or months', dwmtemp,temporal).
object(h,'changes in tscale of asymmetric cooling', actemp,temporal).
object(h,'changes in tscale of loss of axial clr', lactemp,temporal).
object(h,'temporal changes for friction control', fctemp,temporal).
object(h,'changes in tscale of shorted turn', sttemp,temporal).
object(h,'changes in tscale of sticking windings', swtemp,temporal).
object(h,'changes in tscale of step',step,temporal).
object(h,'changes in tscale of rub',rtemp,temporal).
object(h,'changes on a low pressure turbine', lp,location).
object(h,'changes at turbine end of machine', turbend,location).
object(h,'change at generator end of the machine', genend,location).
object(h,'changes pred at turbine end of machine', predturbend,location).
object(h,'changes almost exclusively on the hp rotor', hp_excl,location).
object(h,'change pred at generator end of the machine', predgenend,location).
object(h,'changes on other than the hp rotor', hp_oth,location).
object(h,'changes on several bearings', brgseveral,location).
object(h,'changes on a few bearings',brgfew,location).
object(h,'changes on more than one bearing', brggrtl,location).
object(h,'changes on one bearing only',brgone,location).
object(h,'changes over the full speed range',srf,speed).
object(h,'changes over a bit of the speed range', srabit,speed).
object(h,'changes over a larger speed range', srlarger,speed).
object(h,'changes over a large speed range', srlarge,speed).
object(h,'changes predominantly in once per rev', oprpred,freq).
object(h,'changes in all frequency components',apr,freq).
object(h,'changes in synchronous frequency components', synch,freq).
```

The 'imp' predicate:

The 'imp' predicate represents knowledge for refutation of hypotheses as explained in chapter 4 on knowledge representation and inference.

Clauses of the 'imp' predicate are of the form:

imp(ATOM,LIST-OF-OBJECTS,N,P)

where:

1) ATOM is an identifier which describes an object which is either a fault or a 'high level symptom'.

2) LIST-OF-OBJECTS is a list of objects which is interpreted as a logical disjunction of those objects.

3) NSUP is the necessary support that the object represented by ATOM will result in one or more of the OBJECTS in the list LIST-OF-OBJECTS.

4) PSUP is the necessary support that the object represented by ATOM will result in one or more of the OBJECTS in the list LIST-OF-OBJECTS. PSUP is always set to 1.0 (the reason for this is explained in chapter 4 on knowledge representation and inference.

The above clause represents the following support logic rule:

ATOM ---> disjunction of LIST-OF-OBJECTS : [N,P]

This is explained fully in chapter 4.

The imp predicate is as follows:

```
imp(predturbend,[turbend],1,1).
imp(predgenend,[genend],1,1).
imp(turbend,[hp,ip,lp],1,1).
imp(genend,[exc,gen],1,1).
imp(lp,[lp1,lp2,lp3],1,1).
imp(oilinbore,[oibtemp],1,1).
imp(lossaxclr,[lactemp],1,1).
imp(ascooling,[actemp],1,1).
imp(shturn,[sttemp],1,1).
imp(fteostw,[ru,rd],1,1).
imp(synch,[opr,tpr],1,1).
imp(apr,[subsynch],1,1).
imp(apr,[tpr],1,1).
imp(apr,[opr],1,1).
imp(oprpred,[opr],1,1).
imp(ascooling,[rc2,rc2swh],1,1).
imp(ascooling,[oprpred],1,1).
imp(ascooling,[genend],1,1).
imp(lossaxclr,[rc2,rc2swh],1,1).
imp(lossaxclr,[oprpred],1,1).
imp(lossaxclr,[genend],1,1).
imp(oilinbore,[oprpred],1,1).
imp(oilinbore,[turbend],1,1).
imp(oilinbore,[ruelrest],1,1).
imp(shturn,[rc2,rc2swh],1,1).
imp(shturn,[mfd],1,1).
imp(shturn,[oprpred],1,1).
imp(shturn,[genend],1,1).
imp(stwinds,[genend],1,1).
imp(stwinds,[oprpred],1,1).
imp(mm,[oob],1,1).
imp(oob,[oprpred],1,1).
imp(lm,[step],1,1).
imp(lm,[oob],1,1).
imp(rub,[oprpred],1,1).
```

```
imp(locloose,[brgone],1,1).
imp(align,[synch],1,1).
imp(permbend,[oprpred],1,1).
imp(permbend,[perm],1,1).
imp(permbend,[pbindev],1,1).
imp(trotcrack,[cssmcs],1,1).
imp(trotcrack,[brggrt1],1,1).
imp(stas,[ssmcs],1,1).
imp(crack,[stas],1,1).
imp(st,[lcs],0.9,1).
imp(ascooling,[brgseveral],0.75,1).
imp(lossaxclr,[brgseveral],0.75,1).
imp(oilinbore,[brgseveral],0.75,1).
imp(shturn,[brgseveral],0.75,1).
imp(rub,[brgseveral],0.75,1).
imp(loose,[brgfew],0.75,1).
imp(align,[pedtilt],0.75,1).
imp(align,[jackop],0.75,1).
imp(align,[srcor12],0.75,1).
imp(permbend,[brgseveral],0.75,1).
imp(trotcrack,[prog],0.75,1).
imp(stas,[sfsr],0.7,1).
imp(locloose,[lostm],0.6,1).
imp(loose,[lostm],0.6,1).
imp(crack,[tprtrend],0.6,1).
imp(rub,[rtemp],0.5,1).
imp(ascooling,[srlarger],0.5,1).
imp(lossaxclr,[srlarger],0.5,1).
imp(oilinbore,[srlarger],0.5,1).
imp(rub,[srlarger],0.5,1).
imp(permbend,[srlarger],0.5,1).
imp(permbend,[rpmslecc],0.5,1).
imp(oob,[brgseveral],0.25,1).
imp(oob,[srlarge],0.25,1).
imp(locloose,[apr],0.25,1).
imp(loose,[apr],0.25,1).
imp(trotcrack,[srf],0.25,1).
imp(align,[dpviecc],0.1,1).
imp(align,[brgbrc],0.1,1).
imp(align,[ccs],0.1,1).
imp(srabit,[exgstemp,mfd],0.6,1).
imp(srabit,[rd],0.4,1).
imp(any_old_thing,[oprpred,srlarge,brgseveral],0.8,1).
imp(stwinds,[ru],0.2,1).
imp(stwinds,[brgseveral],0.4,1).
imp(stwinds,[srlarger],0.4,1).
imp(stwinds,[swtemp],0.2,1).
imp(trotcrack,[vsthens],0.4,1).
```

The 'desc' predicate:

The 'desc' clauses represent knowledge about what type of knowledge is relevant to corroborating certain fault hypotheses, and what is the relatve importance of these different types to each fault.

The terms which are used in the following are fully explained in chapter 4 on knowledge representation and inference.

Clauses of the 'desc' predicate are of the form:

desc(ATOM,LIST-OF-LISTS,N,P)

where:

1) ATOM represents a fault

2) LIST-OF-LISTS represents a LIST of LISTS. Each sublist represents the importance of a certain type of description in corroborating the fault. Each sublist is of the form:

[DESC,ND,PD]

where DESC is the name of a type of description and
ND represents the threshold value of the description DESC. PD is always set to 1.0.

3) N and P represent the necessary and possible support that all the descriptions DESC in LIST-OF-LISTS constitute a definitive description of the fault represented by ATOM.

Consider an example. The clause:

desc(lm,[[oob,0.3,1],[temporal,0.4,1]],1,1).

represents that the fault 'lm' which stands for 'loss of' mass (see the earlier 'object' predicate) is corroborated with necessary support 1.0, by the component descriptions 'oob' and 'temporal'.

The threshold values for 'oob' and 'temporal' are 0.3 and 0.4 respectively.

The 'desc' predicate is as follows:

desc(lm,[[oob,0.3,1],[temporal,0.4,1]],1,1).
desc(test_object,[[freq,0.4,1],[oob,0,1]],0.8,1).
desc(predgenend,[[predgenend,0,1]],1,1).

```
desc(predturbend,[[predturbend,0,1]],1,1).
desc(lp,[[lp,0,1]],1,1).
desc(genend,[[genend,0,1]],1,1).
desc(turbend,[[turbend,0,1]],1,1).
desc(swhirl,[[freq,0,1],[temporal,0.7,1],[eoh,0.4,1], [location,0.2,1]],0,1).
desc(fricon,[[fricon,0,1]],1,1).
desc(oob,[[oob,0,1]],1,1).
desc(fteostw,[[fteostw,0,1]],1,1).
desc(apr,[[apr,0,1]],1,1).
desc(synch,[[synch,0,1]],1,1).
desc(crack,[[trendrel,0.7,1],[stas,0.2,1]],0.7,1).
desc(oprpred,[[oprpred,0,1]],1,1).
desc(lossaxclr,[[speed,0.3,1],[location,0.2,1],
[freq,0.6,1],[temporal,0.2,1],[eoh,0.5,1]],0.4,1).
desc(oilinbore,[[temporal,0.3,1],[eoh,0.4,1],
[freq,0.8,1],[location,0.2,1],[speed,0.4,1]],1,1).
desc(ascooling,[[eoh,0.4,1],[temporal,0.5,1],
[freq,0.8,1],[location,0.1,1],[speed,0.2,1]],1,1).
desc(permbend,[[freq,0.8,1],[location,0.3,1],
[speed,0.6,1],[temporal,0.1,1],[eoh,0.6,1]],1,1).
desc(rub,[[eoh,0.7,1],[temporal,0.4,1],[freq,0.8,1],[location,0.3,1],[speed,0.2,1]],1,1).
desc(mm,[[fricon,0.4,1],[eoh,0.5,1],[oob,0.5,1]],0.9,1).
desc(stwinds,[[eoh,0.4,1],[temporal,0.2,1],[freq,0.8,1],[location,0.3,1],[speed,0.3,1]],1,
1).
desc(shturn,[[eoh,0.1,1],[nonvib,0.8,1],[temporal,0.3,1],[freq,0.5,1],[location,0.2,1],[sp
eed,0.7,1]],1,1).
desc(trotcrack,[[eoh,0.8,1],[speed,0.1,1], [location,0.2,1],[temporal,0.5,1]],1,1).
desc(loose,[[location,0,1],[freq,0,1],[nonvib,0,1]],1,1).
desc(locloose,[[nonvib,0.9,1],[freq,0.3,1], [location,0.1,1]],1,1).
desc(align,[[nonvib,0.4,1],[freq,0.5,1], [location,0.5,1],[speed,0.3,1]],1,1).
```

The 'measure' predicate:

The 'measure' predicate represents knowledge used in corroborating fault hypotheses and are accessed via the 'desc' predicate which is described above.

Clauses of the 'measure' predicate are of the form:

measure(ATOM,DESC,N,LIST-OF-OBJECTS,NS,PS)

where:

1) ATOM is the name of a fault, to which this 'measure' is relevant.

2) DESC is the name of a component description such as 'oob' which stands for 'out of balance'.

3) N is simply the number of this clause (within the set of clauses relevant to ATOM).

4) LIST-OF-OBJECTS is a list of objects which when taken in conjunction have the potential to assign a necessary support of NS to the DESC aspect of the fault ATOM.

The above is explained in full in chapter 4 on knowledge representation and inference.

The measure predicate is as follows:

```
measure(trotcrack,stas,1,[ssmcs,sfsr],1,1).
measure(trotcrack,stas,2,[sfsr],0.7,1).
measure(trotcrack,stas,3,[ssmcs],0.4,1).
measure(trotcrack,stas,4,[],0,1).
measure(trotcrack,st,1,[ccs],0.8,1).
measure(trotcrack,st,2,[],0,1).
measure(any,oob,1,[oprpred,srlarge,brgseveral],1,1).
measure(any,oob,2,[srlarge,brgseveral],0.5,1).
measure(any,oob,3,[oprpred,brgseveral],0.7,1).
measure(any,oob,4,[brgseveral],0.2,1).
measure(any,oob,6,[srlarge],0.2,1).
measure(any,oob,7,[oprpred],0.5,1).
measure(lm,temporal,1,[step],1,1).
measure(lm,temporal,2,[],0,1).
measure(mm,eoh,1,[rd,ru],1,1).
measure(mm,eoh,2,[ru],1,1).
measure(mm,eoh,3,[rd],1,1).
measure(mm,eoh,4,[],0,1).
measure(any,fricon,1,[prog,smstrsteps,fctemp],1,1).
measure(any,fricon,2,[fctemp],0.6,1).
measure(any,fricon,3,[prog,smstrsteps],0.9,1).
measure(any,fricon,4,[],0,1).
measure(permbend,eoh,1,[pbindev],1,1).
measure(permbend,eoh,2,[],0,1).
measure(permbend,temporal,1,[perm],1,1).
measure(permbend,temporal,2,[],0,1).
measure(permbend,speed,1,[rpmslecc,srlarger],1,1).
measure(permbend,speed,2,[srlarger],0.6,1).
measure(permbend,speed,3,[rpmslecc],0.7,1).
measure(permbend,speed,4,[],0,1).
measure(permbend,location,1,[brgseveral],1,1).
measure(permbend,location,2,[],0,1).
measure(permbend,freq,1,[oprpred],1,1).
measure(permbend,freq,2,[],0,1).
measure(shturn,freq,1,[oprpred],1,1).
measure(shturn,freq,2,[],0,1).
measure(shturn,nonvib,1,[mfd],1,1).
measure(shturn,nonvib,2,[],0,1).
measure(ascooling,speed,1,[srlarger],1,1).
measure(ascooling,speed,2,[],0,1).
measure(ascooling,location,1,[predgenend,brgseveral],1,1).
measure(ascooling,location,2,[brgseveral],0.6,1).
measure(ascooling,location,3,[predgenend],0.2,1).
measure(ascooling,location,4,[],0,1).
```

```
measure(ascooling,freq,1,[oprpred],1,1).
measure(ascooling,freq,2,[],0,1).
measure(stwinds,speed,1,[srlarger],1,1).
measure(stwinds,speed,2,[],0,1).
measure(stwinds,location,1,[predgenend,brgseveral],1,1).
measure(stwinds,location,2,[brgseveral],0.3,1).
measure(stwinds,location,3,[predgenend],0.4,1).
measure(stwinds,location,4,[],0,1).
measure(stwinds,freq,1,[oprpred],1,1).
measure(stwinds,freq,2,[],0,1).
measure(stwinds,eoh,1,[revrd,fteostw],1,1).
measure(stwinds,eoh,2,[fteostw],0.5,1).
measure(stwinds,eoh,3,[revrd],0.7,1).
measure(stwinds,eoh,4,[],0,1).
measure(lossaxclr,speed,1,[srlarger],1,1).
measure(lossaxclr,speed,2,[],0,1).
measure(lossaxclr,location,1,[predgenend,brgseveral],1,1).
measure(lossaxclr,location,2,[brgseveral],0.2,1).
measure(lossaxclr,location,3,[predgenend],0.4,1).
measure(lossaxclr,location,4,[],0,1).
measure(lossaxclr,freq,1,[oprpred],1,1).
measure(lossaxclr,freq,2,[],0,1).
measure(trotcrack,location,1,[brgseveral],1,1).
measure(trotcrack,location,2,[],0,1).
measure(trotcrack,speed,1,[ccs,cssmcs,srf],1,1).
measure(trotcrack,speed,2,[cssmcs,srf],0.9,1).
measure(trotcrack,speed,3,[ccs,srf],0.3,1).
measure(trotcrack,speed,4,[srf],0.2,1).
measure(trotcrack,speed,5,[ccs,cssmcs],0.7,1).
measure(trotcrack,speed,6,[cssmcs],0.5,1).
measure(trotcrack,speed,7,[ccs],0.15,1).
measure(trotcrack,speed,8,[],0,1).
measure(oilinbore,speed,1,[srlarger],1,1).
measure(oilinbore,speed,2,[],0,1).
measure(oilinbore,location,1,[brgseveral,predturbend],1,1).
measure(oilinbore,location,2,[predturbend],0.2,1).
measure(oilinbore,location,3,[brgseveral],0.5,1).
measure(oilinbore,location,4,[],0,1).
measure(oilinbore,freq,1,[oprpred],1,1).
measure(oilinbore,freq,2,[],0,1).
measure(oilinbore,eoh,1,[ruelrest],1,1).
measure(oilinbore,eoh,2,[],0,1).
measure(rub,speed,1,[srlarge],1,1).
measure(rub,speed,2,[],0,1).
measure(rub,location,1,[brgseveral],1,1).
measure(rub,location,2,[],0,1).
measure(rub,freq,1,[oprpred],1,1).
measure(rub,freq,2,[],0,1).
measure(rub,eoh,1,[toofru],1,1).
measure(rub,eoh,2,[exgstemp],1,1).
measure(rub,eoh,3,[],0,1).
measure(locloose,location,1,[brgone],1,1).
measure(locloose,location,2,[],0,1).
measure(locloose,freq,1,[apr],1,1).
measure(locloose,freq,2,[],0,1).
measure(locloose,nonvib,1,[lostm],1,1).
measure(locloose,nonvib,2,[],0,1).
measure(loose,location,1,[brgfew],1,1).
```

```
measure(loose,location,2,[],0,1).
measure(loose,freq,1,[apr],1,1).
measure(loose,freq,2,[],0,1).
measure(loose,nonvib,1,[lostm],1,1).
measure(loose,nonvib,2,[],0,1).
measure(align,speed,1,[srcor12,ccs],1,1).
measure(align,speed,2,[ccs],0.3,1).
measure(align,speed,3,[srcor12],0.7,1).
measure(align,speed,4,[],0,1).
measure(align,location,1,[brgbrc,brgfew],1,1).
measure(align,location,2,[brgfew],0.5,1).
measure(align,location,3,[brgbrc],0.7,1).
measure(align,location,4,[],0,1).
measure(align,freq,1,[subsynch,synch],1,1).
measure(align,freq,2,[synch],0.8,1).
measure(align,freq,3,[subsynch],0.8,1).
measure(align,freq,4,[],0,1).
measure(align,nonvib,1,[dpviecc,jackop,pedtilt],1,1).
measure(align,nonvib,2,[jackop,pedtilt],0.5,1).
measure(align,nonvib,3,[dpviecc,pedtilt],0.3,1).
measure(align,nonvib,4,[pedtilt],0.2,1).
measure(align,nonvib,5,[dpviecc,jackop],0.4,1).
measure(align,nonvib,6,[jackop],0.2,1).
measure(align,nonvib,7,[dpviecc],0.2,1).
measure(align,nonvib,8,[],0,1).
measure(srf,apr,_1959,[srper],_1961,_1962).
measure(brgfew,_1958,_1959,[brgnof],_1961,_1962).
measure(any,mtos,1,[medium],1,1).
measure(any,mtos,2,[slow],1,1).
measure(any,mtos,3,[],0,1).
measure(any,ftos,1,[fast],1,1).
measure(any,ftos,2,[medium],1,1).
measure(any,ftos,3,[slow],1,1).
measure(any,ftos,4,[],0,1).
measure(srlarge,apr,_1959,[srper],_1961,_1962).
measure(brgseveral,_1958,_1959,[brgnof],_1961,_1962).
measure(srlarger,_1958,_1959,[srper],_1961,_1962).
measure(brgone,_1958,_1959,[brgnof],_1961,_1962).
measure(shturn,location,1,[brgseveral,predgenend],1,1).
measure(shturn,location,2,[predgenend],0.3,1).
measure(shturn,location,3,[brgseveral],0.2,1).
measure(shturn,location,4,[],0,1).
measure(shturn,eoh,1,[rc2swh],1,1).
measure(ascooling,eoh,1,[rc2],1,1).
measure(ascooling,eoh,2,[],0,1).
measure(trotcrack,eoh,1,[rc2],1,1).
measure(trotcrack,eoh,2,[],0,1).
measure(medium,apr,_1959,[time],_1961,_1962).
measure(slow,_1958,_1959,[time],_1961,_1962).
measure(vslow,_1958,_1959,[time],_1961,_1962).
measure(step,_1958,_1959,[time],_1961,_1962).
measure(fast,_1958,_1959,[time],_1961,_1962).
measure(lossaxclr,eoh,1,[rc2swh],1,1).
measure(lossaxclr,eoh,2,[],0,1).
measure(oilinbore,temp,1,[],0,1).
measure(any,apr,1,[subsynch,tpr,opr],1,1).
measure(any,apr,2,[tpr,opr],0.7,1).
measure(any,apr,3,[subsynch,opr],0.7,1).
```

```
measure(any,apr,4,[opr],0.3,1).
measure(any,apr,5,[subsynch,tpr],0.7,1).
measure(any,apr,6,[tpr],0.3,1).
measure(any,apr,7,[subsynch],0.1,1).
measure(any,apr,8,[],0,1).
measure(any,synch,1,[opr],1,1).
measure(any,synch,2,[tpr],1,1).
measure(any,synch,3,[],0,1).
measure(any,fteostw,1,[ru],1,1).
measure(any,fteostw,2,[rd],1,1).
measure(any,fteostw,3,[],0,1).
measure(any,oprpred,1,[fairlynottpr,opr],1,1).
measure(any,oprpred,2,[],0,1).
measure(brggrt1,_1958,_1959,[brgnof],_1961,_1962).
measure(lossaxclr,temporal,1,[lactemp],1,1).
measure(lossaxclr,temporal,2,[],0,1).
measure(ascooling,temporal,1,[actemp],1,1).
measure(ascooling,temporal,2,[],0,1).
measure(rub,temporal,1,[rtemp],0.3,1).
measure(rub,temporal,2,[rtemp,oprerat],1,1).
measure(rub,temporal,3,[rtemp,prog],1,1).
measure(rub,temporal,4,[rtemp,cyc],1,1).
measure(rub,temporal,5,[],0,1).
measure(stwinds,temporal,1,[smstrsteps,swtemp],1,1).
measure(stwinds,temporal,2,[swtemp],1,1).
measure(stwinds,temporal,3,[smstrsteps],0.3,1).
measure(stwinds,temporal,4,[],0,1).
measure(shturn,temporal,1,[sttemp],1,1).
measure(shturn,temporal,2,[],0,1).
measure(trotcrack,temporal,1,[prog,dwmtemp],1,1).
measure(trotcrack,temporal,2,[dwmtemp],0.3,1).
measure(trotcrack,temporal,3,[prog],0.3,1).
measure(trotcrack,temporal,4,[],0,1).
measure(align,temporal,1,[dwmtemp],1,1).
measure(align,temporal,2,[],0,1).
measure(loose,temporal,1,[dwmtemp],1,1).
measure(loose,temporal,2,[],0,1).
measure(locloose,temporal,1,[dwmtemp],1,1).
measure(locloose,temporal,2,[],0,1).
measure(lactemp,bum,_1959,[time],_1961,_1962).
measure(oilinbore,temporal,1,[oibtemp],1,1).
measure(oilinbore,temporal,2,[],0,1).
measure(oibtemp,_1958,_1959,[time],_1961,_1962).
measure(actemp,_1958,_1959,[time],_1961,_1962).
measure(rtemp,_1958,_1959,[time],_1961,_1962).
measure(swtemp,_1958,_1959,[time],_1961,_1962).
measure(sttemp,_1958,_1959,[time],_1961,_1962).
measure(dwmtemp,_1958,_1959,[time],_1961,_1962).
measure(shturn,eoh,2,[rc2],1,1).
measure(shturn,eoh,3,[],0,1).
measure(shturn,speed,1,[srlarger],1,1).
measure(shturn,speed,2,[],0,1).
measure(fctemp,_1958,_1959,[time],_1961,_1962).
measure(swhirl,location,1,[fairlynotip,hp],1,1).
measure(swhirl,location,2,[hp],0.7,1).
measure(swhirl,location,3,[fairlynotip],0.000999996,1).
measure(swhirl,location,4,[],0,1).
measure(swhirl,eoh,1,[loaddep,loadrev],1,1).
```

```
measure(swhirl,eoh,2,[loaddep],0.5,1).
measure(swhirl,eoh,3,[],0,1).
measure(swhirl,temporal,1,[step],1,1).
measure(swhirl,temporal,2,[prog],1,1).
measure(swhirl,temporal,3,[],0,1).
measure(swhirl,freq,1,[subsynch],1,1).
measure(swhirl,freq,2,[],0,1).
measure(any,lp,1,[lp1],1,1).
measure(any,lp,2,[lp2],1,1).
measure(any,lp,3,[lp3],1,1).
measure(any,lp,4,[],0,1).
measure(any,genend,1,[exc],1,1).
measure(any,genend,2,[gen],1,1).
measure(any,genend,3,[],0,1).
measure(any,predgenend,1,[fairlynotturbend,genend],1,1).
measure(any,predgenend,2,[genend],0.5,1).
measure(any,predgenend,3,[fairlynotturbend],0.2,1).
measure(any,predgenend,4,[],0,1).
measure(any,predturbend,1,[fairlynotgenend,turbend],1,1).
measure(any,predturbend,2,[turbend],0.5,1).
measure(any,predturbend,3,[fairlynotgenend],0.2,1).
measure(any,predturbend,4,[],0,1).
measure(any,oob,5,[oprpred,srlarge],0.8,1).
measure(any,oob,10,[],0,1).
measure(any,oob,9,[],0,1).
measure(any,oob,8,[],0,1).
measure(morning,_1958,_1959,[hour],_1961,_1962).
measure(afternoon,_1958,_1959,[hour],_1961,_1962).
measure(coffee_time,_1958,_1959,[hour],_1961,_1962).
measure(any,turbend,1,[hp],1,1).
measure(any,turbend,2,[ip],1,1).
measure(any,turbend,3,[lp],0.2,1).
measure(any,turbend,4,[],0,1).
```

The 'rdmatch' predicate:

The 'rdmatch' predicate represents knowledge about rannge of values of the domain attributes of fuzzy sets. The use of fuzzy sets in the expert system is fully described in chapter 4 on knowledge representation and inference.

Clauses of the 'rdmatch' predicate are of the form:

object(TEXT,ATOM,RANGE)

where:

1) TEXT is a textual description of an object which is the domain attribute of afuxxy set, and ATOM is a corresponding itentifier.

2) RANGE is the range of permissible values for the object ATOM.

The 'rdmatch' predicate is as follows:

```
rdmatch('% of speed range',srper,0,100).
rdmatch('number of bearings',brgnof,0,14).
rdmatch(time,time,-20,20).
rdmatch('pounds sterling',pounds,0,1000000).
```

The 'dmatch' predicate:

The 'dmatch' predicate represents knowledge about fuzzy sets, the use of which is fully described in chapter 4 on knowledge representation and inference.

Clauses of the 'dmatch' predicate are of the form:

dmatch(RANGE-ATOM,DOMAIN-ATOM)

where:

1) RANGE-ATOM is the name of an object which is a fuzzy set.

2) DOMAIN-ATOM is the name of an object which is the domain attribute of the fuzzy relation for the object RANGE-ATOM.

For example, the clause:

dmatch(srf,srper) represents that 'srf' (changes in vibrations over the whole speed range) is a fuzzy relation on 'srper' (the percentage of the spped range over which there has been a change in vibrations).

The dmatch predicate is as follows:

```
dmatch(fctemp,time).
dmatch(step,time).
dmatch(oibtemp,time).
dmatch(sttemp,time).
dmatch(actemp,time).
dmatch(ltemp,time).
dmatch(dwmtemp,time).
dmatch(lactemp,time).
dmatch(rtemp,time).
dmatch(swtemp,time).
dmatch(brgseveral,brgnof).
dmatch(brgfew,brgnof).
dmatch(brggrtl,brgnof).
dmatch(brgone,brgnof).
dmatch(srf,srper).
```

dmatch(srlarge,srper).
dmatch(srlarger,srper).

Predicates which represent fuzzy relations:

The following 16 predicates represent fuzzy realtions.

The clauses in each predicate are of the form:

PRED-NAME(N1,N2)

where:

1) PRED-NAME is the name of a fuzzy set.

2) N1 is a value of the domain attribute of the fuzzy set (which is determined by looking at the 'dmatch' predicate.

3) N2 is the membership level of N1 to the fuzzy set PRED-NAME.

For example, the clause:

srf(40,0.25) represents that the membership level of '40 percent of the speed range' to the set 'the full speed range' is 0.25.

The use of fuzzy sets in the expert system is described in chapter 4 on knowledge represetation.

The 'fuzzy set' predicates are as follows:

step(-(20),1).
step(-(2.3),1).
step(0,0).
step(20,0).

lactemp(0,0).
lactemp(4.09,0).
lactemp(4.79,1).
lactemp(6.40,1).
lactemp(7.09,0).
lactemp(20,0).

```
actemp(0,0).
actemp(4.09,0).
actemp(4.79,1).
actemp(6.40,1).
actemp(7.50,0).
actemp(20,0).

fctemp(0,0).
fctemp(5.7,0).
fctemp(6.40,1).
fctemp(20,1).

rtemp(0,0).
rtemp(5.7,0).
rtemp(6.40,1).
rtemp(8.7,1).
rtemp(9.10,0).
rtemp(20,0).

oibtemp(A,B):-
      rtemp(A,B).

swtemp(A,B):-
      lactemp(A,B).

sttemp(A,B):-
      actemp(A,B).

dwmtemp(0,0).
dwmtemp(10,0).
dwmtemp(11.37,1).
dwmtemp(20,1).

brgseveral(0,0).
brgseveral(1,0).
brgseveral(2,0.5).
brgseveral(3,1).
brgseveral(4,0.75).
brgseveral(5,0.5).
brgseveral(6,0.25).
brgseveral(7,0).
brgseveral(14,0).

brgfew(0,0).
brgfew(1,0).
brgfew(2,1).
brgfew(3,0.75).
brgfew(4,0.25).
brgfew(5,0.1).
brgfew(6,0).
brgfew(14,0).

brggrt1(0,0).
brggrt1(1,0).
brggrt1(2,1).
brggrt1(14,1).
```

```
brgone(0,0).
brgone(1,1).
brgone(2,0).
brgone(14,0).

srf(0,0).
srf(10,0).
srf(20,0).
srf(30,0.1).
srf(40,0.25).
srf(50,0.4).
srf(60,0.75).
srf(70,0.9).
srf(80,1).
srf(100,1).

srlarger(0,0).
srlarger(20,0.05).
srlarger(30,0.15).
srlarger(40,0.35).
srlarger(50,0.6).
srlarger(60,0.9).
srlarger(70,1).
srlarger(100,1).

srlarge(0,0).
srlarge(10,0).
srlarge(20,0.1).
srlarge(30,0.25).
srlarge(40,0.5).
srlarge(50,0.8).
srlarge(60,1).
srlarge(100,1).
```

The 'data' predicate:

The 'data' predicate does not represent expert knowledge. Instead, it represents information about recent and current machine behaviour.

Clauses of the 'data' predicate are of the form:

data(ATOM,TYPE,N,P)

where:

1) ATOM is the name of an object.

2) N and P are the necessary and possible supports for the object ATOM.

3) TYPE describes the origin of evidence relating to the object ATOM. If TYPE = 'neg' then this indicates that the evidence represented by the support pair [N,P]

was infered by negative testing. If TYPE = 'pos' then this indicates that the evidence represented by the support pair [N,P] was infered by positive testing. If TYPE = X (where X is a PROLOG variable of the form _nnn) then this indicates that the evidence represented by the support pair [N,P] was infered by a combination of positive and negative testing or was input directly by a user.

Example 'data' clauses are as follows:

```
data(oprerat,_109,0,0).
data(subsynch,_109,0,0).
data(tpr,_109,1,1).
data(opr,_109,0.8,1).
data(jackop,_109,0,0).
data(pedtilt,_109,0,0).
data(lostm,_109,0,0).
data(mfd,_109,0,0).
data(exgstemp,_109,0,0).
data(fteostw,_109,0,0).
data(dpviecc,_109,0,0).
data(ruelrest,_109,0,0).
data(pbindev,_109,0,0).
data(toofru,_109,0,0).
data(rd,_109,0,0).
data(ru,_109,0,0).
data(revrd,_109,0,0).
data(rc2swh,_109,0,0).
data(rc2,_109,0,0).
data(brgnof,_109,4,4).
data(exc,_109,0.4,0.4).
data(gen,_109,1,1).
data(lp3,_109,0.3,0.3).
data(lp2,_109,0.2,0.2).
data(lp1,_109,0,0).
data(ip,_109,0,0).
data(hp,_109,0,0).
data(brgbrc,_109,0,0).
data(smstrsteps,A,0,0).
data(fthenstep,A,0,0).
data(perm,A,0,0).
data(vsthens,A,1,1).
data(prog,A,1,1).
data(cyc,A,0,0).
data(time,A,15.1045,15.1045).
data(cssmcs,A,1,1).
data(ccs,A,1,1).
data(srper,A,60,80).
data(rpmslecc,A,0,1).
data(srcor12,A,0.5,0.5).
data(brggrt1,A,1.0,1.0).
data(srf,A,0.75,1.0).
data(trotcrack,neg,0,1.0).
data(brgseveral,A,1.0,1.0).
data(dwmtemp,A,1.0,1.0).
data(trotcrack,pos,0.6,1.0).
data(trotcrack,full,0.6,1.0).
```

APPENDIX B

SAMPLE OUTPUT GENERATED BY THE EXPERT SYSTEM

This appendix contains annotated sample output from the expert system, generated by two, user sessions.

The sample output covers all aspects of the expert system.

In the following two sections, the line:

********** CLEAR SCREEN **********

indicates a point at which the terminal screen would be cleared. Text enclosed in a starred box is given as an explanation of the output, and would not actually appear on the terminal screen:

```
**************************************************
*                                                *
*          Text enclosed in a starred box is     *
*          for explanation only.                 *
*                                                *
**************************************************
```

Sample output session number 1.

```
**************************************************
*    The expert system is activated by           *
*    entering PROLOG with a saved state from     *
*    an earlier session. This obviates           *
*    the need to load several large PROLOG       *
*    files each time expert system is run.       *
*                                                *
*    When PROLOG has loaded, the user types      *
*    'go', and the system presents an 'ok : '    *
*    prompt.          The user then types 'shell'*
*    and the expert system checks its            *
*    knowledge base for syntax errors and        *
*    internal inconsistencies, and then          *
*    displays its main menu                      *
**************************************************
```

```
% /usr/local/bin/nip -P200 -C1024 system

Edinburgh Prolog  (version 1.3)
AI Applications Institute, Edinburgh University,  11-Apr-86

| ?- go.
ok : shell
checking knowledge base
completed check of knowledge base

********************** CLEAR SCREEN ********************
*************************************************************
*    The user selects option 1 on this menu,
*    and then instructs the system to
*    create a text file which contains an
*    an explanation of the knowledge relating
*    to a single fault. The user returns to
*    the main menu by pressing return as many
*    times as necessary.
*************************************************************

expert

                                                  page 1
1       [make up statement of kb]
2       [edit knowledge]
3       [generate hypotheses]
4       [test hypotheses]
5       [evaluate remedial actions]
6       [reset diagnosis]
7       [backup expert system]
:1
:

********************** CLEAR SCREEN ********************

expert                  statement

                                                  page 1
1       [construct statements for each fault]
2       [make up a specified statement for printing]
3       [make up a full statement for printing]
4       [send the statement to the line printer]
5       [exit to shell]
:2
:
do you want the statement to include
an explanation of its layout : y

********************** CLEAR SCREEN ********************

select faults to be included in the statement
                                                  page 1
1       [h,alignment change,align,fault]
2       [h,looseness,loose,fault]
3       [h,localised looseness,locloose,fault]
4       [h,non axisymmetric cracking,trotcrack,fault]
```

```
5       [h,shorted turn,shturn,fault]
6       [h,sticking windings,stwinds,fault]
7       [h,loss of mass,lm,fault]
8       [h,movement of mass,mm,fault]
9       [h,rub,rub,fault]
10      [h,permanent bend,permbend,fault]
:2
:
********** CLEAR SCREEN **********

expert                  statement

                                        page 1
1       [construct statements for each fault]
2       [make up a specified statement for printing]
3       [make up a full statement for printing]
4       [send the statement to the line printer]
5       [exit to shell]
:

********** CLEAR SCREEN **********

**************************************************
*   The user selects option 2 from the main      |
*   menu in order to edit the knowledge base.    |
**************************************************

expert

                                        page 1
1       [make up statement of kb]
2       [edit knowledge]
3       [generate hypotheses]
4       [test hypotheses]
5       [evaluate remedial actions]
6       [reset diagnosis]
7       [backup expert system]
:2
:

********** CLEAR SCREEN **********

expert                  edit

                                        page 1
1       [edit knowledge about an object]
2       [remove knowledge about an object]
3       [edit help knowledge]
4       [edit symptoms]
5       [save a file]
:1
:

********** CLEAR SCREEN **********
```

```
**************************************************
*    The user adds a new object to the existing  |
*    list of objects.                            |
**************************************************
```

select object to edit

page 1

```
1       [h,alignment change,align,fault]
2       [h,looseness,loose,fault]
3       [h,localised looseness,locloose,fault]
4       [h,non axisymmetric cracking,trotcrack,fault]
5       [h,shorted turn,shturn,fault]
6       [h,sticking windings,stwinds,fault]
7       [h,loss of mass,lm,fault]
8       [h,movement of mass,mm,fault]
9       [h,rub,rub,fault]
10      [h,permanent bend,permbend,fault]
:a1
:new  > [h,example fault,ef,fault]
```

********** CLEAR SCREEN **********

```
**************************************************
*    The user selects the new object in order    |
*    to edit its associated knowledge.           |
*    However, the system recognises that there   |
*    is no existing knowledge and so begins      |
*    a knowledge acquisition session.            |
**************************************************
```

select object to edit

page 1

```
1       [h,alignment change,align,fault]
2       [h,example fault,ef,fault]
3       [h,looseness,loose,fault]
4       [h,localised looseness,locloose,fault]
5       [h,non axisymmetric cracking,trotcrack,fault]
6       [h,shorted turn,shturn,fault]
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
11      [h,permanent bend,permbend,fault]
:2
:
```

do you want to enter knowledge in
the form of rules or in the form
of a relation (rules/rel) ----- > rules

********** CLEAR SCREEN **********

```
**************************************************
*    The user has specified that knowledge is
*    rule based so the system asks questions
*    to determine the necessary details.
*
*    The user first describes the fault by
*    selecting a number of complimentary
*    descriptions from the list given. In
*    the example below, only one description
*    is selected.
*
*    Knowledge acquisition continues until
*    all knowledge has been input to the system.
**************************************************
```

```
select all relevent descriptions of :
example fault

                                                page 1
1       [specific,out of balance,oob]
2       [generic,e, oc, or h related,eoh]
3       [generic,non-vibrational,nonvib]
4       [specific,friction controled,fricon]
5       [generic,location,location]
6       [generic,frequency,freq]
7       [generic,speed,speed]
8       [generic,temporal,temporal]
9       [generic,time of day,hour]
:1
:

-------------------------------------------------------

for the description ** out of ballance **
of the object ** example fault **
assign importance

    0.0 not important
    1.0 not very important
    2.0 fairly important
    3.0 important
    4.0 very important
    5.0 vital

N = 4
```

********** CLEAR SCREEN **********

** out of balance
e, oc, or h related
non-vibrational
friction controled
location
frequency
speed
temporal
time of day

how useful are the starred descriptions
(when taken together in context), as
a definition of ** example fault

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 3

********** CLEAR SCREEN **********

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1

1 [changes on several bearings,brgseveral]
2 [changes predominantly in once per rev,oprpred]
3 [changes over a large speed range,srlarge]
:1
:2
:3
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
changes over a large speed range;
changes predominantly in once per rev;
changes on several bearings.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

```
N = 5

********** CLEAR SCREEN **********

expert                    edit

                                                    page 1
1       [edit knowledge about an object]
2       [remove knowledge about an object]
3       [edit help knowledge]
4       [edit symptoms]
5       [save a file]
:2
:

********** CLEAR SCREEN **********

**************************************************
*    Having just added knowledge relating to     *
*    a new fault 'example fault', the user       *
*    decides to remove it. In response, the      *
*    system deletes all associated knowledge.    *
**************************************************

select object to remove

                                                    page 1
1       [h,alignment change,align,fault]
2       [h,example fault,ef,fault]
3       [h,looseness,loose,fault]
4       [h,localised looseness,locloose,fault]
5       [h,non axisymmetric cracking,trotcrack,fault]
6       [h,shorted turn,shturn,fault]
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
:2
:

********** CLEAR SCREEN **********

**************************************************
*    The user returns to the main menu and       *
*    reselects the knowledge editor.             *
**************************************************

expert                    edit

                                                    page 1
1       [edit knowledge about an object]
2       [remove knowledge about an object]
3       [edit help knowledge]
4       [edit symptoms]
5       [save a file]
:1
:
```

********** CLEAR SCREEN **********

****
* The user now decides to re-enter knowledge
* about 'example fault'. This time the user
* selects several complimentary descriptions
* of the fault and goes through a lengthy
* question and answer session in which he
* describes knowledge for deriving supporting
* and opposing evidence of 'example fault'.
****

select object to edit

page 1

1 [h,alignment change,align,fault]
2 [h,looseness,loose,fault]
3 [h,localised looseness,locloose,fault]
4 [h,non axisymmetric cracking,trotcrack,fault]
5 [h,shorted turn,shturn,fault]
6 [h,sticking windings,stwinds,fault]
7 [h,loss of mass,lm,fault]
8 [h,movement of mass,mm,fault]
9 [h,rub,rub,fault]
10 [h,permanent bend,permbend,fault]
:a1
:new > [h,example fault,ef,fault]

********** CLEAR SCREEN **********

select object to edit

page 1

1 [h,alignment change,align,fault]
2 [h,example fault,ef,fault]
3 [h,looseness,loose,fault]
4 [h,localised looseness,locloose,fault]
5 [h,non axisymmetric cracking,trotcrack,fault]
6 [h,shorted turn,shturn,fault]
7 [h,sticking windings,stwinds,fault]
8 [h,loss of mass,lm,fault]
9 [h,movement of mass,mm,fault]
10 [h,rub,rub,fault]
11 [h,permanent bend,permbend,fault]
:2
:

do you want to enter knowledge in
the form of rules or in the form
of a relation (rules/rel) ----- > rules

********** CLEAR SCREEN **********

select all relevent descriptions of :
example fault

page 1

1 [specific,out of balance,oob]
2 [generic,e, oc, or h related,eoh]
3 [generic,non-vibrational,nonvib]
4 [specific,friction controled,fricon]
5 [generic,location,location]
6 [generic,frequency,freq]
7 [generic,speed,speed]
8 [generic,temporal,temporal]
9 [generic,time of day,hour]
:1
:5
:6
:

---

for the description ** frequency **
of the object ** example fault **
assign importance

0.0 not important
1.0 not very important
2.0 fairly important
3.0 important
4.0 very important
5.0 vital

N = 3

for the description ** location **
of the object ** example fault **
assign importance

0.0 not important
1.0 not very important
2.0 fairly important
3.0 important
4.0 very important
5.0 vital

N = 5

for the description ** out of ballance **
of the object ** example fault **
assign importance

0.0 not important
1.0 not very important
2.0 fairly important
3.0 important
4.0 very important
5.0 vital

N = 4

********** CLEAR SCREEN **********

** out of balance
e, oc, or h related
non-vibrational
friction controled
** location
** frequency
speed
temporal
time of day

how useful are the starred descriptions
(when taken together in context), as
a definition of ** example fault

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 4

********** CLEAR SCREEN **********

select conjuncts to measure :
example fault (frequency)
previously no symptoms used to measure :
example fault (frequency)

page 1

1 [l,erratic behaviour of once per rev response,
oprerat,freq]
2 [l,changes in sub-synchronous response,
subsynch,freq]
3 [l,change in response at twice per revolution,
tpr,freq]
4 [l,change in response at once per revolution,
opr,freq]
5 [h,changes predominantly in once per rev,
oprpred,freq]

```
6       [h,changes in all frequency components,apr,freq]
7       [h,changes in synchronous frequency components,
            synch,freq]
:1
:3
:
```

how "useful" is the rule below
example fault (frequency) :-
        erratic behaviour of once per rev response,
        change in response at twice per revolution.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 5

how "useful" is the rule below
example fault (frequency) :-
        change in response at twice per revolution.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 3

how "useful" is the rule below
example fault (frequency) :-
        erratic behaviour of once per rev response.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 2

********** CLEAR SCREEN **********

select conjuncts to measure :
        example fault (location)
previously no symptoms used to measure :
    example fault (location)


```
1       [l,recip changes at opp ends of a brg boat,
            brgbrc,location]
2       [l,number of bearings,brgnof,location]
```

```
3       [l,changes on the high pressure turbine,
            hp,location]
4       [l,changes on the intermediate pressure turbine,
            ip,location]
5       [l,changes on the low pressure 1 turbine,
            lp1,location]
6       [l,changes on the low pressure 2 turbine,
            lp2,location]
7       [l,changes on the low pressure 3 turbine,
            lp3,location]
8       [l,changes on the generator rotor,gen,location]
9       [l,changes on the exciter,exc,location]
10      [h,changes on a low pressure turbine,lp,location]
:3

:2
:

how "useful" is the rule below
example fault ( location ) :-
        changes on the high pressure turbine,
        number of bearings.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 5

how "useful" is the rule below
example fault ( location ) :-
        number of bearings.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 3

how "useful" is the rule below
example fault ( location ) :-
        changes on the high pressure turbine.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 3
```

********** CLEAR SCREEN **********

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1

1 [change in response at twice per revolution,tpr]
2 [erratic behaviour of once per rev response,oprerat]
:1
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
change in response at twice per revolution.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 2

********** CLEAR SCREEN **********

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1

1 [erratic behaviour of once per rev response,oprerat]
:1
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
erratic behaviour of once per rev response.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful

4.0 very useful
5.0 superb

N = 4

********** CLEAR SCREEN **********

```
**************************************************
*    This is the section where the user enters   |
*    knowledge for deriving opposing evidence    |
*    of 'example fault'.                         |
**************************************************
```

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1

1 [number of bearings,brgnof]
2 [changes on the high pressure turbine,hp]
:1
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
number of bearings.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 4

********** CLEAR SCREEN **********

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1

1 [changes on the high pressure turbine,hp]
:1
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
changes on the high pressure turbine.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 0

********** CLEAR SCREEN **********

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1

1 [changes on several bearings,brgseveral]
2 [changes predominantly in once per rev,oprpred]
3 [changes over a large speed range,srlarge]
:1
:2
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
changes predominantly in once per rev;
changes on several bearings.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 5

********** CLEAR SCREEN **********

generating rules for refutation of :
example fault
select disjuncts to use as the body of a rule :
{ <object> ===> <body> :[1,1] } or
{ <object> :- NOT <body> :[0,1-1] }

page 1
1 [changes over a large speed range,srlarge]
:1
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
changes over a large speed range.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 2

********** CLEAR SCREEN **********

expert edit

page 1
1 [edit knowledge about an object]
2 [remove knowledge about an object]
3 [edit help knowledge]
4 [edit symptoms]
5 [save a file]
:

********** CLEAR SCREEN **********

****
* Knowledge elicitation for 'example fault'
* has now been completed, and the user returns
* to the expert system's main menu.
*
* The user recusrively calls the expert system
* to demonstrate its flexibility.
****

expert

page 1
1 [make up statement of kb]
2 [edit knowledge]

```
3       [generate hypotheses]
4       [test hypotheses]
5       [evaluate remedial actions]
6       [reset diagnosis]
7       [backup expert system]
:
ok : shell
checking knowledge base
completed check of knowledge base

********** CLEAR SCREEN **********

**************************************************
*    Once again, the user selects the knowledge
*    editor.
**************************************************
expert

                                        page 1
1       [make up statement of kb]
2       [edit knowledge]
3       [generate hypotheses]
4       [test hypotheses]
5       [evaluate remedial actions]
6       [reset diagnosis]
7       [backup expert system]
:2
:

********** CLEAR SCREEN **********

expert                  edit

                                        page 1
1       [edit knowledge about an object]
2       [remove knowledge about an object]
3       [edit help knowledge]
4       [edit symptoms]
5       [save a file]
:1
:

********** CLEAR SCREEN **********

**************************************************
*    The user decides to edit knowledge about the
*    recently added 'example fault'.
**************************************************
select object to edit

                                        page 1
1       [h,alignment change,align,fault]
2       [h,example fault,ef,fault]
3       [h,looseness,loose,fault]
4       [h,localised looseness,locloose,fault]
5       [h,non axisymmetric cracking,trotcrack,fault]
6       [h,shorted turn,shturn,fault]
```

```
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
:2
:

********** CLEAR SCREEN **********

**************************************************
*   The user has the option of editing rules
*   for deriving supporting or oposing evidence.
*
*   The user decides to edit former.
**************************************************

expert              edit        edit object rules

editing rules for:
        example fault
                                                page 1
1       [edit positive rules]
2       [edit negative rules]
:1
:

********** CLEAR SCREEN **********

**************************************************
*   The current description of the fault is
*   shown in tabular form. This includes the
*   threshold value T_l of each description,
*   and the extent O_l to which the overall
*   description is definitive of the fault.
*
*   The user decides to add a new complimentary
*   description.
**************************************************

expert              edit        edit object rules
edit object pos

[ T_l , T_u ]:        N       DESC ( example fault )
--------------------------------------------------
[0.400, 1.000] :      1       frequency
[0.000, 1.000] :      2       location
[0.200, 1.000] :      3       out of ballance
--------------------------------------------------
[0.800, 1.000] :   =    [O_l,O_u]
--------------------------------------------------
                                                page 1
1       [add a new description]
2       [modify a description]
3       [modify outer description]
4       [remove a description]
5       [edit the measures of a description]
:1
:
```

********** CLEAR SCREEN **********

```
**************************************************
*    The user decides that the fault can also be
*    described in terms of speed related
*    symptoms.
**************************************************
```

```
[ T_l ,  T_u ]:        N       DESC ( example fault )
-------------------------------------------------------
[0.400, 1.000] :       1       freq
[0.000, 1.000] :       2       location
[0.200, 1.000] :       3       out of ballance
-------------------------------------------------------
[0.800, 1.000] :   =    [O_l,O_u]
-------------------------------------------------------
select additional description for example fault
                                        page 1
1       [specific,out of balance,oob]
2       [generic,e, oc, or h related,eoh]
3       [generic,non-vibrational,nonvib]
4       [specific,friction controled,fricon]
5       [generic,location,location]
6       [generic,frequency,freq]
7       [generic,speed,speed]
8       [generic,temporal,temporal]
9       [generic,time of day,hour]
:7
:
```

```
**************************************************
*    The user specifies the importance, and
*    therefore the threshold value, of the
*    new description when applied to the
*    fault.
**************************************************
```

```
enter value for importance of
description ** speed

    0.0 not important
    1.0 not very important
    2.0 fairly important
    3.0 important
    4.0 very important
    5.0 vital

N = 3
```

********** CLEAR SCREEN **********

```
**************************************************
*    A new overall assessment of the descriptions
*    ir required.
**************************************************
```

** out of balance
e, oc, or h related
non-vibrational
friction controled
** location
** frequency
** speed
temporal
time of day

how useful are the starred descriptions
(when taken together in context), as
a definition of ** example fault

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 5

********** CLEAR SCREEN **********

****
* The user returns to the 'edit object pos'
* menu and then goes on to demonstrate how
* the other menu options work.
****

expert edit edit object rules
edit object pos

[T_l , T_u]: N DESC (example fault)

[0.400, 1.000] : 1 speed
[0.400, 1.000] : 2 frequency
[0.000, 1.000] : 3 location
[0.200, 1.000] : 4 out of ballance

[1.000, 1.000] : = [O_l,O_u]

---

page 1

1 [add a new description]
2 [modify a description]
3 [modify outer description]
4 [remove a description]
5 [edit the measures of a description]
:2
:
description (N) : 2

enter value for importance of
description ** frequency

0.0 not important
1.0 not very important
2.0 fairly important
3.0 important
4.0 very important
5.0 vital

N = 4

********** CLEAR SCREEN **********

expert edit edit object rules
edit object pos

[T_l , T_u]: N DESC (example fault)

[0.200, 1.000] : 1 frequency
[0.400, 1.000] : 2 speed
[0.000, 1.000] : 3 location
[0.200, 1.000] : 4 out of ballance

[1.000, 1.000] : = [O_l,O_u]
---

page 1

1 [add a new description]
2 [modify a description]
3 [modify outer description]
4 [remove a description]
5 [edit the measures of a description]
:3
:

********** CLEAR SCREEN **********

** out of balance
e, oc, or h related
non-vibrational
friction controled
** location
** frequency
** speed
temporal
time of day

how useful are the starred descriptions
(when taken together in context), as
a definition of ** example fault

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 4

********** CLEAR SCREEN **********

expert edit edit object rules
edit object pos

[T_l , T_u]: N DESC (example fault)

[0.200, 1.000] : 1 frequency
[0.400, 1.000] : 2 speed
[0.000, 1.000] : 3 location
[0.200, 1.000] : 4 out of ballance

[0.800, 1.000] : = [O_l,O_u]

---

page 1

1 [add a new description]
2 [modify a description]
3 [modify outer description]
4 [remove a description]
5 [edit the measures of a description]
:4
:
Description (N) : 1

********** CLEAR SCREEN **********

** out of balance
e, oc, or h related
non-vibrational
friction controled
** location
frequency
** speed
temporal
time of day

how useful are the starred descriptions
(when taken together in context), as
a definition of ** example fault

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 5

```
********** CLEAR SCREEN **********

expert                  edit      edit object rules
edit object pos

[ T_l ,  T_u ]:       N       DESC ( example fault )
-------------------------------------------------------
[0.400, 1.000] :      1       speed
[0.000, 1.000] :      2       location
[0.200, 1.000] :      3       out of ballance
-------------------------------------------------------
[1.000, 1.000] :  =   [O_l,O_u]
-------------------------------------------------------
                                        page  1
 1       [add a new description]
 2       [modify a description]
 3       [modify outer description]
 4       [remove a description]
 5       [edit the measures of a description]
:5
:
description (N) : 1

********** CLEAR SCREEN **********

expert                  edit      edit object rules
edit object pos          edit measures

editing measures :
example fault ( speed )
                                        page 1
 1       [display measures]
 2       [modify a measure]
 3       [add a measure]
 4       [remove a measure]
 5       [display symptoms involved]
 6       [reinput all measures]
:1
:

press return to continue

********** CLEAR SCREEN **********


expert                  edit          edit object rules
edit object pos          edit measures

editing measures :
example fault ( speed )
                                        page 1
 1       [display measures]
 2       [modify a measure]
 3       [add a measure]
 4       [remove a measure]
 5       [display symptoms involved]
 6       [reinput all measures]
:6
```

:

********** CLEAR SCREEN **********

```
**************************************************
*    The user selects the symptoms which can be  *
*    used to measure the description.            *
**************************************************
```

select conjuncts to measure :
example fault (speed)
previously no symptoms used to measure :
example fault (speed)

page 1

```
1       [l,changes at crit speeds and sub-multiples,
            cssmcs,speed]
2       [l,changes in critical speeds,ccs,speed]
3       [l,percentage of speed range,srper,speed]
4       [l,eccentricity at low rpms,rpmslecc,speed]
5       [l,1 and 2 per rev sr correspondance,
            srcor12,speed]
6       [h,changes over the full speed range,srf,speed]
7       [h,changes over a bit of the speed range,
            srabit,speed]
8       [h,changes over a larger speed range,
            srlarger,speed]
9       [h,changes over a large speed range,
            srlarge,speed]
:1
:2
:3
:
```

```
**************************************************
*    The system suggests all possible rule       *
*    combinations, and the user gives an         *
*    assessment of the rules effectiveness       *
*    in measuring the description.               *
**************************************************
```

how "useful" is the rule below
example fault (speed) :-
changes at crit speeds and sub-multiples,
changes in critical speeds,
percentage of speed range.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 5

how "useful" is the rule below
example fault (speed) :-
changes in critical speeds,
percentage of speed range.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 3

how "useful" is the rule below
example fault (speed) :-
changes at crit speeds and sub-multiples,
percentage of speed range.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 2

how "useful" is the rule below
example fault (speed) :-
percentage of speed range.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 1

how "useful" is the rule below
example fault (speed) :-
changes at crit speeds and sub-multiples,
changes in critical speeds.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = N = 3

how "useful" is the rule below
example fault (speed) :-
changes in critical speeds.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 0

how "useful" is the rule below
example fault (speed) :-
changes at crit speeds and sub-multiples.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 1

********** CLEAR SCREEN **********

expert edit edit object rules
edit object pos edit measures

editing measures :
example fault (speed)
page 1

1 [display measures]
2 [modify a measure]
3 [add a measure]
4 [remove a measure]
5 [display symptoms involved]
6 [reinput all measures]
:1
:
FAULT = example fault
DESC = speed

[1.000, 1.000] : 1
changes at crit speeds and sub-multiples,
changes in critical speeds,
percentage of speed range.

[0.600, 1.000] : 2
changes in critical speeds,
percentage of speed range.

[0.400, 1.000] : 3
changes at crit speeds and sub-multiples,
percentage of speed range.

```
[0.200, 1.000] :        4
        percentage of speed range.
-------------------------------------------------------
[0.600, 1.000] :        5
        changes at crit speeds and sub-multiples,
        changes in critical speeds.
----More--(67%)---------------------------------------
[0.000, 1.000] :        6
        changes in critical speeds.
-------------------------------------------------------
[0.200, 1.000] :        7
        changes at crit speeds and sub-multiples.
-------------------------------------------------------
[0.000, 1.000] :        8
-------------------------------------------------------
```

press return to continue

********** CLEAR SCREEN **********

expert edit edit object rules
edit object pos edit measures

editing measures :
example fault (speed)

page 1

1 [display measures]
2 [modify a measure]
3 [add a measure]
4 [remove a measure]
5 [display symptoms involved]
6 [reinput all measures]
:

********** CLEAR SCREEN **********

expert edit edit object rules
edit object pos

[T_l , T_u]: N DESC (example fault)

```
-------------------------------------------------------
[0.400, 1.000] :        1       speed
[0.000, 1.000] :        2       location
[0.200, 1.000] :        3       out of ballance
-------------------------------------------------------
[1.000, 1.000] :   =    [O_l,O_u]
-------------------------------------------------------
```

page 1

1 [add a new description]
2 [modify a description]
3 [modify outer description]
4 [remove a description]
5 [edit the measures of a description]
:

********** CLEAR SCREEN **********

```
**************************************************
*    Having just edited knowledge for deriving
*    supporting evidence, the user chooses to
*    edit knowledge for deriving opposing
*    evidence.
**************************************************
```

expert edit edit object rules

editing rules for:
example fault

page 1

1 [edit positive rules]
2 [edit negative rules]
:2
:

********** CLEAR SCREEN **********

expert edit edit object rules
edit object neg

[E_l , E_u]: N EFFECT (example fault)

[0.400, 1.000] : 1 change in response at twice per revolution.
[0.800, 1.000] : 2 erratic behaviour of once per rev response.
[0.800, 1.000] : 3 number of bearings.
[1.000, 1.000] : 4 changes predominantly in once per rev;
changes on several bearings.
[0.400, 1.000] : 5 changes over a large speed range.

page 1

1 [add a new effect]
2 [modify an effect]
3 [remove an effect]
:1
:

********** CLEAR SCREEN **********

```
**************************************************
*    The system displays all current rules.
*    Semi-colons denote disjunction.
**************************************************
```

select additional effect (a disjunction) for ef

page 1

1 [l,slower change has small str of steps, smstrsteps,temporal]
2 [l,fast changes followed by a step, fthenstep,temporal]
3 [l,changes permanent since some datum, perm,temporal]
4 [l,very slow followed slow changes,

```
            vsthens,temporal]
5       [l,changes progressive with time,prog,temporal]
6       [l,changes cyclical with time,cyc,temporal]
7       [l,time taken,time,temporal]
8       [l,excursion of gland steam temperature,
            exgstemp,nonvib]
9       [l,magnetic field disruption,mfd,nonvib]
10      [l,jacking oil pressure,jackop,nonvib]
:f

********** CLEAR SCREEN **********

**************************************************
*    The user adds a new rule for deriving       *
*    opposing evidence of 'example fualt'        *
**************************************************

select additional effect (a disjunction) for ef
                                          page 2
1       [l,pedestal tilt,pedtilt,nonvib]
2       [l,lost motion,lostm,nonvib]
3       [l,opposing changes in ped vibs and eccentricity,
            dpviecc,eoh]
4       [l,ft event or op-conds for sticking windings,
            fteostw,eoh]
5       [l,changes in behaviour preceeded by a run-down,
            rd,eoh]
6       [l,changes in behaviour preceeded by a run-up,
            ru,eoh]
7       [l,run-up or early loading restricted,
            ruelrest,eoh]
8       [l,i squared and step with hysterisis,rc2swh,eoh]
9       [l,changes reversible with run-down,revrd,eoh]
10      [l,permanent bend inducing event,pbindev,eoh]
:2
:3
:

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
        opposing changes in ped vibs and eccentricity;
        lost motion.

    0.0 useless
    1.0 not very useful
    2.0 fairly useful
    3.0 useful
    4.0 very useful
    5.0 superb

N = 3
```

********** CLEAR SCREEN **********

expert edit edit object rules
edit object neg

[E_l , E_u]: N EFFECT (example fault)

[0.400, 1.000] : 1 change in response at twice per revolution.
[0.800, 1.000] : 2 erratic behaviour of once per rev response.
[0.800, 1.000] : 3 number of bearings.
[1.000, 1.000] : 4 changes predominantly in once per rev;
changes on several bearings.
[0.400, 1.000] : 5 changes over a large speed range.
[0.600, 1.000] : 6 opposing changes in ped vibs and eccentricity;
lost motion.

page 1

1 [add a new effect]
2 [modify an effect]
3 [remove an effect]
:2
:
effect (N) : 6

how "useful" is the rule below as a basis
for eliminating ** example fault

example fault ===>
opposing changes in ped vibs and eccentricity;
lost motion.

0.0 useless
1.0 not very useful
2.0 fairly useful
3.0 useful
4.0 very useful
5.0 superb

N = 2

********** CLEAR SCREEN **********

expert edit edit object rules
edit object neg

[E_l , E_u]: N EFFECT (example fault)

[0.400, 1.000] : 1 change in response at twice per revolution.
[0.800, 1.000] : 2 erratic behaviour of once per rev response.
[0.800, 1.000] : 3 number of bearings.
[1.000, 1.000] : 4 changes predominantly in once per rev;

changes on several bearings.
[0.400, 1.000] : 5 changes over a large speed range.
[0.400, 1.000] : 6 opposing changes in ped vibs and eccentricity;
lost motion.

page 1

1 [add a new effect]
2 [modify an effect]
3 [remove an effect]
:

********** CLEAR SCREEN **********

expert edit edit object rules

editing rules for:
example fault

page 1

1 [edit positive rules]
2 [edit negative rules]
:

********** CLEAR SCREEN **********

```
**************************************************
*   The user selects an option from the 'edit'   *
*   menu, which enables editing of 'help text    *
*   files'.                                      *
**************************************************
```

expert edit

page 1

1 [edit knowledge about an object]
2 [remove knowledge about an object]
3 [edit help knowledge]
4 [edit symptoms]
5 [save a file]
:3
:

********** CLEAR SCREEN **********

expert edit edit help

page 1

1 [edit terminology (help)]
:1
:

```
********** CLEAR SCREEN **********

expert                  edit                edit help
edit help term

                                            page 1
1       [edit terminology for faults]
2       [edit terminology for descriptions]
3       [edit terminology for symptoms]
:1
:

********** CLEAR SCREEN **********

select faults to edit (help)
                                            page 1
1       [h,alignment change,align,fault]
2       [h,example fault,ef,fault]
3       [h,looseness,loose,fault]
4       [h,localised looseness,locloose,fault]
5       [h,non axisymmetric cracking,trotcrack,fault]
6       [h,shorted turn,shturn,fault]
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
:1
:4
:

text.files/help_fault reconsulted:   2816 bytes      4.78 seconds

********** CLEAR SCREEN **********

editing help for :
        localised looseness
                                            page 1
1       [as for looseness except that changes are
            exhibited]
2       [on one bearing only.]
:

********** CLEAR SCREEN **********

editing help for :
        alignment change
                                            page 1
1       [alignment change refers to a delineation]
2       [of the turbo-generator's rotating shaft]
:

********** CLEAR SCREEN **********

expert                  edit                edit help
edit help term

                                            page 1
1       [edit terminology for faults]
```

```
2       [edit terminology for descriptions]
3       [edit terminology for symptoms]
:

********** CLEAR SCREEN **********

expert                  edit                edit help

                                              page 1
1       [edit terminology (help)]
:

********** CLEAR SCREEN **********

expert                  edit

                                              page 1
1       [edit knowledge about an object]
2       [remove knowledge about an object]
3       [edit help knowledge]
4       [edit symptoms]
5       [save a file]
:1
:

********** CLEAR SCREEN **********

**************************************************
*    The user adds a new object, but this time he
*    specifies the associated knowledge to be
*    relational rather than rule based.
*
*    The system responds by acquiring a fuzzy
*    relation from the user.
**************************************************

select object to edit

                                              page 1
1       [h,alignment change,align,fault]
2       [h,example fault,ef,fault]
3       [h,looseness,loose,fault]
4       [h,localised looseness,locloose,fault]
5       [h,non axisymmetric cracking,trotcrack,fault]
6       [h,shorted turn,shturn,fault]
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
:f

********** CLEAR SCREEN **********

select object to edit

                                              page 2
1       [h,permanent bend,permbend,fault]
2       [h,asymmetric cooling,ascooling,fault]
```

```
3       [h,oil in the bore,oilinbore,fault]
4       [h,loss of axial clearance,lossaxclr,fault]
5       [h,out of ballance,oob,fault]
6       [h,changes in tscale of days weeks or months,
            dwmtemp,temporal]
7       [h,changes in tscale of asymmetric cooling,
            actemp,temporal]
8       [h,changes in tscale of loss of axial clr,
            lactemp,temporal]
9       [h,temporal changes for friction control,
            fctemp,temporal]
10      [h,changes in tscale of shorted turn,
            sttemp,temporal]
:a10
:new  > [h,example relation,er,symptom,temporal]
********** CLEAR SCREEN **********

select object to edit

                                        page 2
1       [h,permanent bend,permbend,fault]
2       [h,asymmetric cooling,ascooling,fault]
3       [h,oil in the bore,oilinbore,fault]
4       [h,loss of axial clearance,lossaxclr,fault]
5       [h,out of ballance,oob,fault]
6       [h,changes in tscale of days weeks or months,
            dwmtemp,temporal]
7       [h,changes in tscale of asymmetric cooling,
            actemp,temporal]
8       [h,changes in tscale of loss of axial clr,
            lactemp,temporal]
9       [h,temporal changes for friction control,
            fctemp,temporal]
10      [h,changes in tscale of shorted turn,
            sttemp,temporal]
11      [h,example relation,er,symptom,temporal]
:11
:
do you want to enter knowledge in
the form of rules or in the form
of a relation (rules/rel) ----- > rel

********** CLEAR SCREEN **********

**************************************************
*    The user specifies the doman object as      *
*    being 'time of day'. The system responds    *
*    by acquiring the supports associated with   *
*    the limiting values of the domain object.   *
*    The system then allows the user to fill in  *
*    the rest of the fuzzy set via the clause    *
*    editor.                                     *
**************************************************
```

```
select domain symptom for example relation

                                                    page 1
1       [% of speed range,srper,0,100]
2       [number of bearings,brgnof,0,14]
3       [time,time,-20,20]
4       [hour of the day,hour,0,24]
5       [pounds sterling,pounds,0,1000000]
6       [length of journey,mileage,0,20000]
7       [duration in minutes,duration in minutes,0,10000]
:4
:

how well does time of day = 0
satisfy the relation example relation
--------- > 0
how well does time of day = 24
satisfy the relation example relation
--------- > 0

********** CLEAR SCREEN **********

editing the relation example relation
tuple : [time of day,<support>]
                                                    page 1
1       [0,0.0]
2       [24,0.0]
:a1
:new  > [6,0.0]

********** CLEAR SCREEN **********

editing the relation example relation
tuple : [time of day,<support>]
                                                    page 1
1       [0,0.0]
2       [6.0,0.0]
3       [24,0.0]
:a2
:new  > [8.5,1.0]

********** CLEAR SCREEN **********

editing the relation example relation
tuple : [time of day,<support>]
                                                    page 1
1       [0,0.0]
2       [6.0,0.0]
3       [8.5,1.0]
4       [24,0.0]
:a3
:new  > [10,0.0]
```

********** CLEAR SCREEN **********

editing the relation example relation
tuple : [time of day,<support>]

page 1

1 [0,0.0]
2 [6.0,0.0]
3 [8.5,1.0]
4 [10.0,0.0]
5 [24,0.0]
:

********** CLEAR SCREEN **********

expert edit

page 1

1 [edit knowledge about an object]
2 [remove knowledge about an object]
3 [edit help knowledge]
4 [edit symptoms]
5 [save a file]
:4
:

********** CLEAR SCREEN **********

****
* The user selects the edit symptoms option
* to demonstrate one method of inputting
* symptoms to the system.
****

expert edit edit symptoms

page 1

1 [input/edit symptoms]
2 [load a file of symptoms]
:1
:

********** CLEAR SCREEN **********

expert edit edit symptoms
modify symptoms

page 1

1 [auto input of symptoms]
2 [edit e, oc, or h related symptoms]
3 [edit non-vibrational symptoms]
4 [edit location symptoms]
5 [edit frequency symptoms]
6 [edit speed symptoms]
7 [edit temporal symptoms]
8 [edit time of day symptoms]
:1
:
remove all current symptoms ? y

********** CLEAR SCREEN **********

```
**************************************************
*    The user chooses the automatic input option.
*    The system responds by requesting input for
*    each symptom, one group at a time.
*
*    The system prompts for the support for each
*    symptom, and the user responds to each one
*    by typing in a either one or two numbers
*    which specify the support pair.
*    In some cases, the user can type y or n or u
*    which correspond to 'yes' 'no' and
*    'uncertain'.
**************************************************
```

```
editing e, oc, or h related symptoms

?  1 [0.000, 1.000] :  opposing changes in ped vibs and
                                    eccentricity
?  2 [0.000, 1.000] :  ft event or op-conds for sticking
                                    windings
?  3 [0.000, 1.000] :  changes in behaviour preceeded by
                                    a run-down
?  4 [0.000, 1.000] :  changes in behaviour preceeded by
                                    a run-up
?  5 [0.000, 1.000] :  run-up or early loading restricted
?  6 [0.000, 1.000] :  i squared and step with hysterisis
?  7 [0.000, 1.000] :  changes reversible with run-down
?  8 [0.000, 1.000] :  permanent bend inducing event
?  9 [0.000, 1.000] :  i squared characteristic
? 10 [0.000, 1.000] :  too fast a run-up

enter symptom number : a
1 >1
2 >8
3 >5
4 >4 6
5 >3 9
6 >u
7 >n
8 >y
9 >4
10 >8
```

********** CLEAR SCREEN **********

```
**************************************************
*    The system redisplays the symptoms with    the
*    new support pairs just supplied by the user.
*
*    Before progressing to the next group of
*    symptoms, the user has the option of
*    revising any number of the supports he has
*    just typed in. He simply types the number
*    of the symptom whose support is incorrect,
*    and the system prompts for a new support.
```

```
**************************************************

editing e, oc, or h related symptoms

*  1 [0.100, 0.100] :  opposing changes in ped vibs and
                                        eccentricity
*  2 [0.800, 0.800] :  ft event or op-conds for sticking
                                        windings
*  3 [0.500, 0.500] :  changes in behaviour preceeded by
                                        a run-down
*  4 [0.400, 0.600] :  changes in behaviour preceeded by
                                        a run-up
*  5 [0.300, 0.900] :  run-up or early loading restricted
?  6 [0.000, 1.000] :  i squared and step with hysterisis
*  7 [0.000, 0.000] :  changes reversible with run-down
*  8 [1.000, 1.000] :  permanent bend inducing event
*  9 [0.400, 0.400] :  i squared characteristic
* 10 [0.800, 0.800] :  too fast a run-up

enter symptom number :

********** CLEAR SCREEN **********

editing non-vibrational symptoms

?  1 [0.000, 1.000] :  excursion of gland steam
                                        temperature
?  2 [0.000, 1.000] :  magnetic field disruption
?  3 [0.000, 1.000] :  jacking oil pressure
?  4 [0.000, 1.000] :  pedestal tilt
?  5 [0.000, 1.000] :  lost motion

enter symptom number : a
1 >2
2 >5 6
3 >7
4 >4 8
5 >u

********** CLEAR SCREEN **********


editing non-vibrational symptoms

*  1 [0.200, 0.200] :  excursion of gland steam
                                        temperature
*  2 [0.500, 0.600] :  magnetic field disruption
*  3 [0.700, 0.700] :  jacking oil pressure
*  4 [0.400, 0.800] :  pedestal tilt
?  5 [0.000, 1.000] :  lost motion

enter symptom number :
```

********** CLEAR SCREEN **********

```
**************************************************
*    The system recognises that the symptom
*    'number of bearings' does not require an
*    appociated support, but instead requires
*    an upper and lower bound (eg. between 2 and
*    five bearings). It therefore delays input
*    until later
**************************************************
```

editing location symptoms

```
?  1 [0.000, 1.000] :  recip changes at opp ends of a brg
                                          boat
?  2 [0.000, 1.000] :  number of bearings
?  3 [0.000, 1.000] :  changes on the high pressure
                                          turbine
?  4 [0.000, 1.000] :  changes on the intermediate
                                          pressure turbine
?  5 [0.000, 1.000] :  changes on the low pressure 1
                                          turbine
?  6 [0.000, 1.000] :  changes on the low pressure 2
                                          turbine
?  7 [0.000, 1.000] :  changes on the low pressure 3
                                          turbine
?  8 [0.000, 1.000] :  changes on the generator rotor
?  9 [0.000, 1.000] :  changes on the exciter

enter symptom number : a
1 >2
2 >-- input delayed --
3 >6
4 >7
5 >u
6 >9
7 >2
8 >2 9
9 >4
-------------------------------------------------
number of bearings
-------------------------------------------------
(h value) (0---14) : 12
```

********** CLEAR SCREEN **********

editing location symptoms

```
*  1 [0.200, 0.200] :  recip changes at opp ends of a brg
                                          boat
*  2 [12.000, 12.000] :  number of bearings
*  3 [0.600, 0.600] :  changes on the high pressure
                                          turbine
*  4 [0.700, 0.700] :  changes on the intermediate
                                          pressure turbine
?  5 [0.000, 1.000] :  changes on the low pressure 1
                                          turbine
```

```
*  6 [0.900, 0.900] :  changes on the low pressure 2
                                        turbine
*  7 [0.200, 0.200] :  changes on the low pressure 3
                                        turbine
*  8 [0.200, 0.900] :  changes on the generator rotor
*  9 [0.400, 0.400] :  changes on the exciter

enter symptom number :

********** CLEAR SCREEN **********

**************************************************
*     The user skips input for some of the
*     groups of symptoms (the user may have
*     insufficient evidence).
**************************************************

editing frequency symptoms

?  1 [0.000, 1.000] :  erratic behaviour of once per rev
                                        response
?  2 [0.000, 1.000] :  changes in sub-synchronous
                                        response
?  3 [0.000, 1.000] :  change in response at twice per
                                        revolution
?  4 [0.000, 1.000] :  change in response at once per
                                        revolution
enter symptom number :

********** CLEAR SCREEN **********

editing speed symptoms

?  1 [0.000, 1.000] :  changes at crit speeds and sub-
                                        multiples
?  2 [0.000, 1.000] :  changes in critical speeds
?  3 [0.000, 1.000] :  percentage of speed range
?  4 [0.000, 1.000] :  eccentricity at low rpms
?  5 [0.000, 1.000] :  1 and 2 per rev sr correspondance

enter symptom number :

********** CLEAR SCREEN **********

editing temporal symptoms

?  1 [0.000, 1.000] :  slower change has small str of
                                        steps
?  2 [0.000, 1.000] :  fast changes followed by a step
?  3 [0.000, 1.000] :  changes permanent since some datum
?  4 [0.000, 1.000] :  very slow followed slow changes
?  5 [0.000, 1.000] :  changes progressive with time
?  6 [0.000, 1.000] :  changes cyclical with time
?  7 [0.000, 1.000] :  time taken

enter symptom number : a
1 >4
```

```
2 >7
3 >4 6
4 >7
5 >3
6 >8
7 >-- input delayed --
---------------------------------------------------
time taken
---------------------------------------------------
(h value) (n unit(S)) : 4 hourd  s

********** CLEAR SCREEN **********

editing temporal symptoms

*  1 [0.400, 0.400] :  slower change has small str of
                                        steps
*  2 [0.700, 0.700] :  fast changes followed by a step
*  3 [0.400, 0.600] :  changes permanent since some datum
*  4 [0.700, 0.700] :  very slow followed slow changes
*  5 [0.300, 0.300] :  changes progressive with time
*  6 [0.800, 0.800] :  changes cyclical with time

********** CLEAR SCREEN **********

expert                  edit          edit symptoms
modify symptoms

                                        page 1
1       [auto input of symptoms]
2       [edit e, oc, or h related symptoms]
3       [edit non-vibrational symptoms]
4       [edit location symptoms]
5       [edit frequency symptoms]
6       [edit speed symptoms]
7       [edit temporal symptoms]
8       [edit time of day symptoms]
:7
:

********** CLEAR SCREEN **********

**************************************************
*    The user chooses to re-edit a single group
*    of symptoms.
**************************************************

editing temporal symptoms

?  1 [0.000, 1.000] :  slower change has small str of
                                        steps
?  2 [0.000, 1.000] :  fast changes followed by a step
?  3 [0.000, 1.000] :  changes permanent since some datum
?  4 [0.000, 1.000] :  very slow followed slow changes
?  5 [0.000, 1.000] :  changes progressive with time
?  6 [0.000, 1.000] :  changes cyclical with time
?  7 [0.000, 1.000] :  time taken
```

```
enter symptom number : 7
-------------------------------------------------------
time taken
-------------------------------------------------------
(h value) (n unit(S)) : 5 hours

********** CLEAR SCREEN **********

editing temporal symptoms

?  1 [0.000, 1.000] :  slower change has small str of
                                              steps
?  2 [0.000, 1.000] :  fast changes followed by a step
?  3 [0.000, 1.000] :  changes permanent since some datum
?  4 [0.000, 1.000] :  very slow followed slow changes
?  5 [0.000, 1.000] :  changes progressive with time
?  6 [0.000, 1.000] :  changes cyclical with time
*  7 [9.798, 9.798] :  time taken

enter symptom number :

********** CLEAR SCREEN **********

**************************************************
*    The user saves the data which has just been
*    input, in a data file.
**************************************************

expert                edit                edit symptoms
modify symptoms

                                              page 1
1       [auto input of symptoms]
2       [edit e, oc, or h related symptoms]
3       [edit non-vibrational symptoms]
4       [edit location symptoms]
5       [edit frequency symptoms]
6       [edit speed symptoms]
7       [edit temporal symptoms]
8       [edit time of day symptoms]
:
save in which file : demo

********** CLEAR SCREEN **********

**************************************************
*    A new data file is loaded. This one was
*    generated by an expert from the CEGB.
**************************************************
```

```
expert                    edit          edit symptoms

                                                page 1
1       [input/edit symptoms]
2       [load a file of symptoms]
:2
:
demo   demo1  demo2  feb10  feb24  feb27  john1  rog1   rog2   wed

file name : rog1


symptoms/rog1 reconsulted:   3904 bytes      6.18 seconds

********** CLEAR SCREEN **********

expert                    edit                  edit symptoms

                                                page 1
1       [input/edit symptoms]
2       [load a file of symptoms]
:

********** CLEAR SCREEN **********

expert                    edit

                                                page 1
1       [edit knowledge about an object]
2       [remove knowledge about an object]
3       [edit help knowledge]
4       [edit symptoms]
5       [save a file]
:

********** CLEAR SCREEN **********

**************************************************
*    The user returns to the main menu and       |
*    decides to test current hypotheses.         |
**************************************************


expert

                                                page 1
1       [make up statement of kb]
2       [edit knowledge]
3       [generate hypotheses]
4       [test hypotheses]
5       [evaluate remedial actions]
6       [reset diagnosis]
7       [backup expert system]
:4
:

********** CLEAR SCREEN **********
```

```
expert                    test

                                                    page 1
1       [perform a negative test of some objects]
2       [perform a positive test of some objects]
3       [perform a full test of some objects]
4       [test hypotheses]
5       [filter hypotheses]
:4
:

********************* CLEAR SCREEN ******************

*****************************************************
*     The user first selects negative testing
*     then positive testing and finally a
*     combined positive and negative testing.
*
*     The user then demonstrates some additional
*     testing options.
*****************************************************

expert                    test          test hypotheses

                                                    page 1
1       [neg diagnosis]
2       [pos diagnosis]
3       [full diagnosis]
:1
:

results of negative diagnosis      :-

[0.000, 1.000] :        movement of mass
[0.000, 1.000] :        non axisymmetric cracking
[0.000, 0.640] :        sticking windings
[0.000, 0.500] :        rub
[0.000, 0.250] :        alignment change
[0.000, 0.175] :        looseness
[0.000, 0.000] :        loss of axial clearance
[0.000, 0.000] :        oil in the bore
[0.000, 0.000] :        asymmetric cooling
[0.000, 0.000] :        permanent bend
[0.000, 0.000] :        loss of mass
[0.000, 0.000] :        shorted turn
[0.000, 0.000] :        localised looseness

press return to continue

********************* CLEAR SCREEN ******************

expert                    test          test hypotheses

                                                    page 1
1       [neg diagnosis]
2       [pos diagnosis]
```

3 [full diagnosis]
:2
:

results of positive diagnosis :-

[0.600, 1.000] : non axisymmetric cracking
[0.260, 1.000] : loss of mass
[0.256, 1.000] : movement of mass
[0.224, 1.000] : rub
[0.107, 1.000] : asymmetric cooling
[0.091, 1.000] : alignment change
[0.051, 1.000] : oil in the bore
[0.050, 1.000] : localised looseness
[0.043, 1.000] : sticking windings
[0.036, 1.000] : permanent bend
[0.016, 1.000] : loss of axial clearance
[0.008, 1.000] : shorted turn
[0.000, 1.000] : looseness

press return to continue

********** CLEAR SCREEN **********

expert test test hypotheses

page 1

1 [neg diagnosis]
2 [pos diagnosis]
3 [full diagnosis]
:3
:

results of full diagnosis :-

[0.600, 1.000] : non axisymmetric cracking
[0.256, 1.000] : movement of mass
[0.126, 0.563] : rub
[0.028, 0.650] : sticking windings
[0.000, 0.250] : alignment change
[0.000, 0.175] : looseness
[0.000, 0.000] : loss of axial clearance
[0.000, 0.000] : oil in the bore
[0.000, 0.000] : asymmetric cooling
[0.000, 0.000] : permanent bend
[0.000, 0.000] : loss of mass
[0.000, 0.000] : shorted turn
[0.000, 0.000] : localised looseness

press return to continue

```
********** CLEAR SCREEN **********

expert                    test        test hypotheses

                                                  page 1
1       [neg diagnosis]
2       [pos diagnosis]
3       [full diagnosis]
:

********** CLEAR SCREEN **********

expert                    test

                                                  page 1
1       [perform a negative test of some objects]
2       [perform a positive test of some objects]
3       [perform a full test of some objects]
4       [test hypotheses]
5       [filter hypotheses]
:1
:
perform neg test of those already pos tested ?

********** CLEAR SCREEN **********

select list of objects

                                                  page 1
1       [h,alignment change,align,fault]
2       [h,example fault,ef,fault]
3       [h,looseness,loose,fault]
4       [h,localised looseness,locloose,fault]
5       [h,non axisymmetric cracking,trotcrack,fault]
6       [h,shorted turn,shturn,fault]
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
:1
:3
:6
:
[0.000, 0.250] :        alignment change
[0.000, 0.175] :        looseness
[0.000, 0.000] :        shorted turn
press return to continue

********** CLEAR SCREEN **********

expert                    test

                                                  page 1
1       [perform a negative test of some objects]
2       [perform a positive test of some objects]
```

```
3       [perform a full test of some objects]
4       [test hypotheses]
5       [filter hypotheses]
:2
:
perform pos test of those already neg tested ? y
[0.600, 1.000] :        non axisymmetric cracking
[0.500, 1.000] :        out of ballance
[0.260, 1.000] :        loss of mass
[0.256, 1.000] :        movement of mass
[0.224, 1.000] :        rub
[0.107, 1.000] :        asymmetric cooling
[0.091, 1.000] :        alignment change
[0.051, 1.000] :        oil in the bore
[0.050, 1.000] :        localised looseness
[0.043, 1.000] :        sticking windings
[0.036, 1.000] :        permanent bend
[0.016, 1.000] :        loss of axial clearance
[0.008, 1.000] :        shorted turn
[0.000, 1.000] :        looseness
press return to continue

********** CLEAR SCREEN **********

expert                  test

                                        page 1
1       [perform a negative test of some objects]
2       [perform a positive test of some objects]
3       [perform a full test of some objects]
4       [test hypotheses]
5       [filter hypotheses]
:resolve

********** CLEAR SCREEN **********

**************************************************
*    Having reset the database by typing         *
*    'resolve', the user switches the on the     *
*    trace which will explain any subsequent     *
*    hypothesis testing.                         *
**************************************************

expert                  test

                                        page 1
1       [perform a negative test of some objects]
2       [perform a positive test of some objects]
3       [perform a full test of some objects]
4       [test hypotheses]
5       [filter hypotheses]
:trace on
```

********** CLEAR SCREEN **********

expert test

page 1

1 [perform a negative test of some objects]
2 [perform a positive test of some objects]
3 [perform a full test of some objects]
4 [test hypotheses]
5 [filter hypotheses]
:3
:
perform full test of those already neg tested ?
perform full test of those already pos tested ?

********** CLEAR SCREEN **********

****
* The trace is shown in action.
****

select list of objects

page 1

1 [h,alignment change,align,fault]
2 [h,example fault,ef,fault]
3 [h,looseness,loose,fault]
4 [h,localised looseness,locloose,fault]
5 [h,non axisymmetric cracking,trotcrack,fault]
6 [h,shorted turn,shturn,fault]
7 [h,sticking windings,stwinds,fault]
8 [h,loss of mass,lm,fault]
9 [h,movement of mass,mm,fault]
10 [h,rub,rub,fault]
:5
:
performing full_test of non axisymmetric cracking :
looking in data for non axisymmetric cracking :
data not found for non axisymmetric cracking
performing R.B.T. of non axisymmetric cracking :
performing neg_test of non axisymmetric cracking :
looking in data for non axisymmetric cracking :
data not found for non axisymmetric cracking
performing R.B.R. of non axisymmetric cracking :
currently non axisymmetric cracking :[0.000,1.000]
proceeding to use the rule ----->
[1.000, 1.000] : non axisymmetric cracking ===>
changes at crit speeds and sub-multiples.
in refutation :
performing neg_test of changes at crit speeds and sub-multiples :
looking in data for changes at crit speeds and sub-multiples :
data found changes at crit speeds and sub-multiples :[1.000,1.000]
completed neg_test changes at crit speeds and sub-multiples :[1.000,1.000]
rule yields non axisymmetric cracking :[0.000,1.000]
combining [0.000, 1.000] : & [0.000, 1.000] : :
gives combined support :[0.000,1.000]
deciding whether to continue refutation
remaining possibility is 1.0 therefore continue refutation

currently non axisymmetric cracking :[0.000,1.000]
proceeding to use the rule ----->
[1.000, 1.000] : non axisymmetric cracking ===>
changes on more than one bearing.
in refutation :
performing neg_test of changes on more than one bearing :
looking in data for changes on more than one bearing :
data not found for changes on more than one bearing
looking in data for number of bearings :
data found number of bearings :[4.000,4.000]
matching with relation changes on more than one bearing
[0.000, 0.000] : number of bearings = 0
[0.000, 0.000] : number of bearings = 1
[1.000, 1.000] : number of bearings = 2
[1.000, 1.000] : number of bearings = 14
relation yeilds changes on more than one bearing :[1.000,1.000]
adding to data [1.000, 1.000] : changes on more than one bearing :
completed neg_test changes on more than one bearing :[1.000,1.000]
rule yields non axisymmetric cracking :[0.000,1.000]
combining [0.000, 1.000] : & [0.000, 1.000] : :
gives combined support :[0.000,1.000]
deciding whether to continue refutation
remaining possibility is 1.0 therefore continue refutation
currently non axisymmetric cracking :[0.000,1.000]
proceeding to use the rule ----->
[0.750, 1.000] : non axisymmetric cracking ===>
changes progressive with time.
in refutation :
performing neg_test of changes progressive with time :
looking in data for changes progressive with time :
data found changes progressive with time :[1.000,1.000]
completed neg_test changes progressive with time :[1.000,1.000]
rule yields non axisymmetric cracking :[0.000,1.000]
combining [0.000, 1.000] : & [0.000, 1.000] : :
gives combined support :[0.000,1.000]
deciding whether to continue refutation
remaining possibility is 1.0 therefore continue refutation
currently non axisymmetric cracking :[0.000,1.000]
proceeding to use the rule ----->
[0.250, 1.000] : non axisymmetric cracking ===>
changes over the full speed range.
in refutation :
performing neg_test of changes over the full speed range :
looking in data for changes over the full speed range :
data not found for changes over the full speed range
looking in data for percentage of speed range :
data found percentage of speed range :[60.000,80.000]
matching with relation changes over the full speed range
[0.000, 0.000] : percentage of speed range = 0
[0.000, 0.000] : percentage of speed range = 10
[0.050, 0.050] : percentage of speed range = 20
[0.100, 0.100] : percentage of speed range = 30
[0.250, 0.250] : percentage of speed range = 40
[0.400, 0.400] : percentage of speed range = 50
[0.750, 0.750] : percentage of speed range = 60
[0.900, 0.900] : percentage of speed range = 70
[1.000, 1.000] : percentage of speed range = 80
[1.000, 1.000] : percentage of speed range = 100

```
relation yeilds changes over the full speed range :[0.750,1.000]
adding to data [0.750, 1.000] : changes over the full speed range :
completed neg_test changes over the full speed range :[0.750,1.000]
rule yields non axisymmetric cracking :[0.000,1.000]
combining [0.000, 1.000] :  &  [0.000, 1.000] :  :
gives combined support  :[0.000,1.000]
deciding whether to continue refutation
remaining possibility is 1.0 therefore continue refutation
currently non axisymmetric cracking :[0.000,1.000]
proceeding to use the rule ----->
[0.400, 1.000] : non axisymmetric cracking ===>
        very slow followed slow changes.
in refutation :
performing neg_test of very slow followed slow changes :
looking in data for very slow followed slow changes :
data found very slow followed slow changes :[1.000,1.000]
completed neg_test very slow followed slow changes :[1.000,1.000]
rule yields non axisymmetric cracking :[0.000,1.000]
combining [0.000, 1.000] :  &  [0.000, 1.000] :  :
gives combined support  :[0.000,1.000]
deciding whether to continue refutation
remaining possibility is 1.0 therefore continue refutation
adding to data [0.000, 1.000] : non axisymmetric cracking :
completed R.B.R. non axisymmetric cracking :[0.000,1.000]
completed neg_test non axisymmetric cracking :[0.000,1.000]
performing pos_test of non axisymmetric cracking :
looking in data for non axisymmetric cracking :
data not found for non axisymmetric cracking
performing R.B.B. of non axisymmetric cracking :
designing a building rule for non axisymmetric cracking :
looking for descriptions of non axisymmetric cracking :
found description of non axisymmetric cracking

[ T_l ,  T_u ] :       N       DESC ( non axisymmetric cracking )
-------------------------------------------------------------
[0.800, 1.000] :       1       e, oc, or h related
[0.100, 1.000] :       2       speed
[0.200, 1.000] :       3       location
[0.500, 1.000] :       4       temporal
-------------------------------------------------------------
[1.000, 1.000] :   =   [O_l,O_u]
-------------------------------------------------------------
choosing description measures of non axisymmetric cracking :
assessing each measure of e, oc, or h related :
assessing measure number 1 :
assessed measure number 1
[1.000, 1.000] : non axisymmetric cracking ( e, oc, or h related ) :-
        i squared characteristic.
assessment is (0.8+(1-0.8)*1)*0 = 0.0
assessing measure number 2 :
assessed measure number 2
[0.000, 1.000] : non axisymmetric cracking ( e, oc, or h related ) :-
        true
assessment is (0.8+(1-0.8)*0)*1 = 0.8
assessing measure number 3 :
measure number 3 not found ------ (no more measures)
assessed each measure of e, oc, or h related
```

```
choosing best measure of e, oc, or h related :
chosen best measure of e, oc, or h related
[0.000, 1.000] : non axisymmetric cracking ( e, oc, or h related ) :-
        true
assessing each measure of speed :
assessing measure number 1 :
assessed measure number 1
[1.000, 1.000] : non axisymmetric cracking ( speed ) :-
        changes in critical speeds,
        changes at crit speeds and sub-multiples,
        changes over the full speed range.
assessment is (0.1+(1-0.1)*1)*0.75 = 0.75
assessing measure number 2 :
assessed measure number 2
[0.900, 1.000] : non axisymmetric cracking ( speed ) :-
        changes at crit speeds and sub-multiples,
        changes over the full speed range.
assessment is (0.1+(1-0.1)*0.9)*0.75 = 0.6825
assessing measure number 3 :
assessed measure number 3
[0.300, 1.000] : non axisymmetric cracking ( speed ) :-
        changes in critical speeds,
        changes over the full speed range.
assessment is (0.1+(1-0.1)*0.3)*0.75 = 0.2775
assessing measure number 4 :
assessed measure number 4
[0.200, 1.000] : non axisymmetric cracking ( speed ) :-
        changes over the full speed range.
assessment is (0.1+(1-0.1)*0.2)*0.75 = 0.21
assessing measure number 5 :
assessed measure number 5
[0.700, 1.000] : non axisymmetric cracking ( speed ) :-
        changes in critical speeds,
        changes at crit speeds and sub-multiples.
assessment is (0.1+(1-0.1)*0.7)*1 = 0.73
assessing measure number 6 :
assessed measure number 6
[0.500, 1.000] : non axisymmetric cracking ( speed ) :-
        changes at crit speeds and sub-multiples.
assessment is (0.1+(1-0.1)*0.5)*1 = 0.55
assessing measure number 7 :
assessed measure number 7
[0.150, 1.000] : non axisymmetric cracking ( speed ) :-
        changes in critical speeds.
assessment is (0.1+(1-0.1)*0.15)*1 = 0.235
assessing measure number 8 :
assessed measure number 8
[0.000, 1.000] : non axisymmetric cracking ( speed ) :-
        true
assessment is (0.1+(1-0.1)*0)*1 = 0.1
assessing measure number 9 :
measure number 9 not found ------ (no more measures)
assessed each measure of speed
choosing best measure of speed :
chosen best measure of speed
```

[1.000, 1.000] : non axisymmetric cracking (speed) :-
changes in critical speeds,
changes at crit speeds and sub-multiples,
changes over the full speed range.
assessing each measure of location : s
assessed each measure of location
choosing best measure of location : s
chosen best measure of location
[1.000, 1.000] : non axisymmetric cracking (location) :-
changes on several bearings.
assessing each measure of temporal : s
assessed each measure of temporal
choosing best measure of temporal : s
chosen best measure of temporal
[1.000, 1.000] : non axisymmetric cracking (temporal) :-
changes progressive with time,
changes in tscale of days weeks or months.
chosen description measures of non axisymmetric cracking
designed a building rule for non axisymmetric cracking
making body of building rule for non axisymmetric cracking :
made body of building rule for non axisymmetric cracking
changes in critical speeds,
changes at crit speeds and sub-multiples,
changes over the full speed range,
changes on several bearings,
changes progressive with time,
changes in tscale of days weeks or months.
assessing newly designed rule
i.e. calculating conditional support pair :
calculating ccs for non axisymmetric cracking (e, oc, or h related)
calculating ccs for non axisymmetric cracking (speed)
calculating ccs for non axisymmetric cracking (location)
calculating ccs for non axisymmetric cracking (temporal)
calculated ccs for non axisymmetric cracking (temporal)
tl + (1 - tl) * ml = 0.5+(1-0.5)*1 = 1.0
calculated ccs for non axisymmetric cracking (location)
tl + (1 - tl) * ml = 0.2+(1-0.2)*1 = 1.0
calculated ccs for non axisymmetric cracking (speed)
tl + (1 - tl) * ml = 0.1+(1-0.1)*1 = 1.0
calculated ccs for non axisymmetric cracking (e, oc, or h related)
tl + (1 - tl) * ml = 0.8+(1-0.8)*0 = 0.8
looking for support for descriptions being definitive :
descriptions are definitive : [Dl,Du] = :[1.000,1.000]
assessed newly designed rule
i.e. calculated conditional support pair :[0.800,1.000]
looking in data for changes in critical speeds :
data found changes in critical speeds :[1.000,1.000]
looking in data for changes at crit speeds and sub-multiples :
data found changes at crit speeds and sub-multiples :[1.000,1.000]
looking in data for changes over the full speed range : s
data found changes over the full speed range :[0.750,1.000]
looking in data for changes on several bearings : s
data found changes on several bearings :[1.000,1.000]
looking in data for changes progressive with time : s
data found changes progressive with time :[1.000,1.000]
looking in data for changes in tscale of days weeks or months : s
data found changes in tscale of days weeks or months :[1.000,1.000]
evaluating building rule

[0.800, 1.000] : non axisymmetric cracking :-

changes in critical speeds,
changes at crit speeds and sub-multiples,
changes over the full speed range,
changes on several bearings,
changes progressive with time,
changes in tscale of days weeks or months.

for non axisymmetric cracking :
evaluating support for conjunction
[1.000, 1.000] : changes in tscale of days weeks or months
[1.000, 1.000] : changes progressive with time
[1.000, 1.000] : changes on several bearings
[1.000, 1.000] : changes at crit speeds and sub-multiples
[1.000, 1.000] : changes in critical speeds
[0.750, 1.000] : changes over the full speed range

evaluated body of rule :[0.750,1.000]
evaluated building rule
< BODY > :[0.750,1.000]
<CONDITIONAL SUPPORT PAIR > :[0.800,1.000]
giving non axisymmetric cracking :[0.600,1.000]
completed R.B.B. non axisymmetric cracking :[0.600,1.000]
completed pos_test non axisymmetric cracking :[0.600,1.000]
combining [0.000, 1.000] : & [0.600, 1.000] : :
gives combined support :[0.600,1.000]
completed R.B.T. non axisymmetric cracking :[0.600,1.000]
completed full_test non axisymmetric cracking :[0.600,1.000]
[0.600, 1.000] : non axisymmetric cracking
press return to continue

********** CLEAR SCREEN **********

****
* The trace is now switched off.
****

expert test

page 1
1 [perform a negative test of some objects]
2 [perform a positive test of some objects]
3 [perform a full test of some objects]
4 [test hypotheses]
5 [filter hypotheses]
:trace off

********** CLEAR SCREEN **********

expert test

page 1
1 [perform a negative test of some objects]
2 [perform a positive test of some objects]
3 [perform a full test of some objects]
4 [test hypotheses]
5 [filter hypotheses]

```
:help

********** CLEAR SCREEN **********

expert                    test                    help

                                                  page 1
1       [help on terminology]
2       [help on use of system]
3       [help on state of diagnosis]
4       [access trace/explanation]
5       [access help utilities]
6       [try out diagnosis with modified symptoms]
:1
:

********** CLEAR SCREEN **********

**************************************************
*    The user show how the help facility works.  |
**************************************************

expert                    test                    help
help_terminology

                                                  page 1
1       [details of a fault]
2       [details of a description]
3       [details of a symptom]
4       [details of the knowledge representation]
5       [details of the inference mechanism]
:1
:

********** CLEAR SCREEN **********

enter list of faults to be explained
                                                  page 1
1       [alignment change]
2       [example fault]
3       [looseness]
4       [localised looseness]
5       [non axisymmetric cracking]
6       [shorted turn]
7       [sticking windings]
8       [loss of mass]
9       [movement of mass]
10      [rub]
:1
:5
:

text.files/help_fault reconsulted:  2816 bytes      4.70 seconds

non axisymmetric cracking describes a region of
discontinuity in the material of the machine's
rotating shaft. The behaviour exhibited is that of
progressive stiffness asymmetry.
```

press return to continue

alignment change refers to a delineation
of the turbo-generator's rotating shaft

press return to continue

********** CLEAR SCREEN **********

expert test help
help_terminology

page 1

1 [details of a fault]
2 [details of a description]
3 [details of a symptom]
4 [details of the knowledge representation]
5 [details of the inference mechanism]
:

********** CLEAR SCREEN **********

expert test help

page 1

1 [help on terminology]
2 [help on use of system]
3 [help on state of diagnosis]
4 [access trace/explanation]
5 [access help utilities]
6 [try out diagnosis with modified symptoms]
:3
:

********** CLEAR SCREEN **********

expert test help
help on diag state

page 1

1 [display objects neg tested so far]
2 [display objects pos tested so far]
3 [display objects full tested so far]
4 [display current symptoms]
5 [display generated rules]
:4
:
[0.000, 0.000] : slower change has small str of steps
[0.000, 0.000] : fast changes followed by a step
[0.000, 0.000] : changes permanent since some datum
[1.000, 1.000] : very slow followed slow changes
[1.000, 1.000] : changes progressive with time
[0.000, 0.000] : changes cyclical with time
[15.104, 15.104] : time taken
[0.000, 0.000] : excursion of gland steam temperature
[0.000, 0.000] : magnetic field disruption

[0.000, 0.000] : jacking oil pressure
[0.000, 0.000] : pedestal tilt
[0.000, 0.000] : lost motion
[0.000, 0.000] : opposing changes in ped vibs and eccentricity
[0.000, 0.000] : ft event or op-conds for sticking windings
[0.000, 0.000] : changes in behaviour preceeded by a run-down
[0.000, 0.000] : changes in behaviour preceeded by a run-up
[0.000, 0.000] : run-up or early loading restricted
[0.000, 0.000] : i squared and step with hysterisis
[0.000, 0.000] : changes reversible with run-down
[0.000, 0.000] : permanent bend inducing event
[0.000, 0.000] : i squared characteristic
[0.000, 0.000] : too fast a run-up
[0.000, 0.000] : erratic behaviour of once per rev response
[0.000, 0.000] : changes in sub-synchronous response
[1.000, 1.000] : change in response at twice per revolution
[0.800, 1.000] : change in response at once per revolution
[0.000, 0.000] : recip changes at opp ends of a brg boat
[4.000, 4.000] : number of bearings
[0.000, 0.000] : changes on the high pressure turbine
[0.000, 0.000] : changes on the intermediate pressure turbine
[0.000, 0.000] : changes on the low pressure 1 turbine
[0.200, 0.200] : changes on the low pressure 2 turbine
[0.300, 0.300] : changes on the low pressure 3 turbine
[1.000, 1.000] : changes on the generator rotor
[0.400, 0.400] : changes on the exciter
[1.000, 1.000] : changes at crit speeds and sub-multiples
[1.000, 1.000] : changes in critical speeds
[60.000, 80.000] : percentage of speed range
[0.000, 1.000] : eccentricity at low rpms
[0.500, 0.500] : 1 and 2 per rev sr correspondance

press return to continue

********** CLEAR SCREEN **********

expert test help
help on diag state

page 1

1 [display objects neg tested so far]
2 [display objects pos tested so far]
3 [display objects full tested so far]
4 [display current symptoms]
5 [display generated rules]
:5
:

********** CLEAR SCREEN **********

select a list of objects

page 1

1 [h,alignment change,align,fault]
2 [h,example fault,ef,fault]
3 [h,looseness,loose,fault]
4 [h,localised looseness,locloose,fault]
5 [h,non axisymmetric cracking,trotcrack,fault]

```
6       [h,shorted turn,shturn,fault]
7       [h,sticking windings,stwinds,fault]
8       [h,loss of mass,lm,fault]
9       [h,movement of mass,mm,fault]
10      [h,rub,rub,fault]
:5
:
[0.800, 1.000] : non axisymmetric cracking :-

        changes in critical speeds,
        changes at crit speeds and sub-multiples,
        changes over the full speed range,
        changes on several bearings,
        changes progressive with time,
        changes in tscale of days weeks or months.
```

```
press return to continue

********** CLEAR SCREEN **********

expert                  test                    help
help on diag state

                                        page 1
1       [display objects neg tested so far]
2       [display objects pos tested so far]
3       [display objects full tested so far]
4       [display current symptoms]
5       [display generated rules]
:
```

```
********** CLEAR SCREEN **********

expert                  test                    help

                                        page 1
1       [help on terminology]
2       [help on use of system]
3       [help on state of diagnosis]
4       [access trace/explanation]
5       [access help utilities]
6       [try out diagnosis with modified symptoms]
:5
:
```

```
********** CLEAR SCREEN **********

expert                  test                    help
help utilities

                                        page 1
1       [abort execution]
2       [break execution]
3       [look at prolog statistics]
4       [start a recursive unix shell]
5       [access unix commands]
```

```
6       [perform visual edit of a file]
7       [load a file]
:
expert                  test                    help

                                                page 1
1       [help on terminology]
2       [help on use of system]
3       [help on state of diagnosis]
4       [access trace/explanation]
5       [access help utilities]
6       [try out diagnosis with modified symptoms]
:

********** CLEAR SCREEN **********

expert                  test

                                                page 1
1       [perform a negative test of some objects]
2       [perform a positive test of some objects]
3       [perform a full test of some objects]
4       [test hypotheses]
5       [filter hypotheses]
:

********** CLEAR SCREEN **********

**************************************************
*    The user ends the session by typing 'abort'
**************************************************

expert

                                                page 1
1       [make up statement of kb]
2       [edit knowledge]
3       [generate hypotheses]
4       [test hypotheses]
5       [evaluate remedial actions]
6       [reset diagnosis]
7       [backup expert system]
:
ok : abort

[Execution aborted]

| ?-
NIP terminated
%
script done on Wed Sep 30 16:14:29 1987
```

Sample output session number 2.

```
**************************************************
*     The user starts another session and goes
*     directly to the 'edit menu'
**************************************************
```

```
Script started on Thu Oct  1 10:06:40 1987
/u% sr/local/bin/nip -P200 -C1024 system

Edinburgh Prolog  (version 1.3)
AI Applications Institute, Edinburgh University,  11-Apr-86

| ?- go.
ok : edit

********** CLEAR SCREEN **********

select object to edit

                                                page 1
1       [h,alignment change,align,fault]
2       [h,looseness,loose,fault]
3       [h,localised looseness,locloose,fault]
4       [h,non axisymmetric cracking,trotcrack,fault]
5       [h,shorted turn,shturn,fault]
6       [h,sticking windings,stwinds,fault]
7       [h,loss of mass,lm,fault]
8       [h,movement of mass,mm,fault]
9       [h,rub,rub,fault]
10      [h,permanent bend,permbend,fault]
:4
:

********** CLEAR SCREEN **********

expert                  edit object rules

editing rules for:
        non axisymmetric cracking
                                                page 1
1       [edit positive rules]
2       [edit negative rules]
:
ok : test
```

********** CLEAR SCREEN **********

```
**************************************************
*    The user selects the 'filter hypotheses'
*    option, which requests support for symptoms
*    as it performs the diagnosis. It requests
*    for support, one symptom at a time, in an
*    order which its calculations indicate will
*    require the minimum number of user inputs.
**************************************************
```

expert test

page 1

1 [perform a negative test of some objects]
2 [perform a positive test of some objects]
3 [perform a full test of some objects]
4 [test hypotheses]
5 [filter hypotheses]
:rf rog1

symptoms/rog1 reconsulted: 3904 bytes 6.29 seconds

********** CLEAR SCREEN **********

expert test

page 1

1 [perform a negative test of some objects]
2 [perform a positive test of some objects]
3 [perform a full test of some objects]
4 [test hypotheses]
5 [filter hypotheses]
:5
:

********** CLEAR SCREEN **********

[0.000, 1.000] : loss of axial clearance
[0.000, 1.000] : oil in the bore
[0.000, 1.000] : asymmetric cooling
[0.000, 1.000] : permanent bend
[0.000, 1.000] : rub
[0.000, 1.000] : movement of mass
[0.000, 1.000] : loss of mass
[0.000, 1.000] : sticking windings
[0.000, 1.000] : shorted turn
[0.000, 1.000] : looseness
[0.000, 1.000] : localised looseness
[0.000, 1.000] : alignment change
[0.000, 1.000] : non axisymmetric cracking

********** CLEAR SCREEN **********

USED----[0.800, 1.000] : change in response at once per revolution

ELIMINATED----loss of axial clearance
ELIMINATED----oil in the bore
ELIMINATED----asymmetric cooling
ELIMINATED----permanent bend
ELIMINATED----loss of mass
ELIMINATED----shorted turn
ELIMINATED----looseness
ELIMINATED----localised looseness
ELIMINATED----alignment change

[0.000, 1.000] : non axisymmetric cracking
[0.000, 0.640] : sticking windings
[0.000, 1.000] : movement of mass
[0.000, 0.500] : rub

********** CLEAR SCREEN **********

USED----[0.800, 1.000] : change in response at once per revolution
USED----[4.000, 4.000] : number of bearings

ELIMINATED----loss of axial clearance
ELIMINATED----oil in the bore
ELIMINATED----asymmetric cooling
ELIMINATED----permanent bend
ELIMINATED----loss of mass
ELIMINATED----shorted turn
ELIMINATED----looseness
ELIMINATED----localised looseness
ELIMINATED----alignment change

[0.000, 1.000] : non axisymmetric cracking
[0.000, 0.640] : sticking windings
[0.000, 1.000] : movement of mass
[0.000, 0.500] : rub

********** CLEAR SCREEN **********

USED----[0.800, 1.000] : change in response at once per revolution
USED----[4.000, 4.000] : number of bearings
USED----[0.400, 0.400] : changes on the exciter

ELIMINATED----loss of axial clearance
ELIMINATED----oil in the bore
ELIMINATED----asymmetric cooling
ELIMINATED----permanent bend

ELIMINATED----loss of mass
ELIMINATED----shorted turn
ELIMINATED----looseness
ELIMINATED----localised looseness
ELIMINATED----alignment change

[0.000, 1.000] : non axisymmetric cracking
[0.000, 0.640] : sticking windings
[0.000, 1.000] : movement of mass
[0.000, 0.500] : rub

********** CLEAR SCREEN **********

USED----[0.800, 1.000] : change in response at once per revolution
USED----[4.000, 4.000] : number of bearings
USED----[0.400, 0.400] : changes on the exciter
USED----[1.000, 1.000] : changes on the generator rotor

ELIMINATED----loss of axial clearance
ELIMINATED----oil in the bore
ELIMINATED----asymmetric cooling
ELIMINATED----permanent bend
ELIMINATED----loss of mass
ELIMINATED----shorted turn
ELIMINATED----looseness
ELIMINATED----localised looseness
ELIMINATED----alignment change

[0.000, 1.000] : non axisymmetric cracking
[0.000, 0.640] : sticking windings
[0.000, 1.000] : movement of mass
[0.000, 0.500] : rub

****
* The user now demonstrates that it is possible
* to short cut the menu system, and input commands
* directly.
****

Function (h for help)? a

[Execution aborted]

| ?- go.
ok : ?lm
neg evidence :[0.000,0.000]
pos evidence :[0.260,1.000]
full evidence :[0.000,0.000]
ok : ?temporal

********** CLEAR SCREEN **********

editing temporal symptoms

* 1 [0.000, 0.000] : slower change has small str of
steps
* 2 [0.000, 0.000] : fast changes followed by a step
* 3 [0.000, 0.000] : changes permanent since some datum
* 4 [1.000, 1.000] : very slow followed slow changes
* 5 [1.000, 1.000] : changes progressive with time
* 6 [0.000, 0.000] : changes cyclical with time
* 7 [15.104, 15.104] : time taken

enter symptom number :
ok : ?opr
old support : 0.800 1.000
change to : 0.6 1
ok : ?opr
old support : 0.600 1.000
change to :
ok : remedial
ok : remedy

********** CLEAR SCREEN **********

expert test remedial

page 1

1 [display significant faults]
2 [display uncertain data]
3 [evaluate uncertainties in data]
:
ok : ptest lm
loss of mass :[0.260,1.000]

ok : ntest lm
loss of mass :[0.000,0.000]

ok : ftest lm
loss of mass :[0.000,0.000]

ok : help temporal

text.files/help_description reconsulted: 1920 bytes 3.29 seconds

within the context of a particular fault: the description
temporal represents a particular group of symptoms that
provide corroborative evidence for the fault.

press return to continue

ok : help oob

text.files/help_fault reconsulted: 2816 bytes 4.85 seconds

press return to continue

```
ok : edit help oob

text.files/help_fault reconsulted:   2816 bytes      4.75 seconds

********************* CLEAR SCREEN ******************

editing help for :
        out of ballance
                                                page 1
1       [no text]
:m1

:old  > [no text]
:new  > [oob is an identifier for out of balance]

********************* CLEAR SCREEN ******************

editing help for :
        out of ballance
                                                page 1
1       [oob is an identifier for out of balance]
:
ok : abort

[Execution aborted]

| ?-
NIP terminated
%
script      done      on      Thu      Oct              1      10:20:19      1987
```


UNIVERSITY
OF BRISTOL
LIBRARY

RING